

NASA Contractor Report 3315

NASA  
CR  
3315  
c.1

Segmentation, Dynamic Storage, and  
Variable Loading on CDC Equipment

Sherwood H. Tiffany

CONTRACT NAS1-16000  
SEPTEMBER 1980

**NASA**

LOAN COPY  
AFWIL TECHNICAL  
KIRTLAND AFB

0062078



TECH LIBRARY KAFB, NM



## NASA Contractor Report 3315

# Segmentation, Dynamic Storage, and Variable Loading on CDC Equipment

Sherwood H. Tiffany  
*Kentron International, Inc.*  
*Hampton, Virginia*

Prepared for  
Langley Research Center  
under Contract NAS1-16000



National Aeronautics  
and Space Administration

**Scientific and Technical  
Information Branch**

1980



# TABLE OF CONTENTS

|  | Page |
|--|------|
| SUMMARY.....   | 1    |
| INTRODUCTION.....  | 1    |
| SECTION I: PROBLEM DESCRIPTION.....                        | 2    |
| SECTION II: PROBLEM SOLUTION.....                          | 3    |
| Load Structures.....                                       | 3    |
| Illustration 1. Tree or Branching Diagram.....             | 3    |
| Data Requirements for Each Branch.....                     | 4    |
| End-of-Code Addresses.....                                 | 4    |
| Programming Steps.....                                     | 4    |
| SECTION III: ILLUSTRATIONS OF TECHNIQUES.....              | 5    |
| Illustration 2. Outline of Sample Program.....             | 5    |
| Branches and Program Code for Interactive Case.....        | 7    |
| Illustration 3. Negative Indexing.....                     | 8    |
| Branches and Program Code for Batch Case.....              | 9    |
| Load Directives.....                                       | 10   |
| Example 1. Load Directives for Interactive Load            | 10   |
| Example 2. Load Directives for Batch Load.....             | 10   |
| Editing Commands for Conversion of Load Directives....     | 11   |
| Example 3. Editing Commands for Conversion of              |      |
| Load Directives from Batch Load to                         |      |
| Interactive Load.....                                      | 11   |
| Editing Commands to Create Address File.....               | 11   |
| SECTION IV: RESULTS.....                                   | 16   |
| Illustration 4. Field Lengths Used for Batch               |      |
| and Interactive Runs.....                                  | 16   |
| CONCLUDING REMARKS.....                                    | 17   |
| APPENDIX A: SAMPLE PROGRAM LISTING.....                    | 18   |
| APPENDIX B: SEGLOAD DIRECTIVES AND TREE DIAGRAMS FOR BATCH |      |
| AND INTERACTIVE RUNS.....                                  | 33   |
| APPENDIX C: LOAD MAP FOR INTERACTIVE RUN.....              | 37   |
| APPENDIX D: ADDRESS FILE FOR INTERACTIVE RUN.....          | 48   |
| APPENDIX E: EDITING FILE TO GENERATE ADDRESSES FROM LOAD   |      |
| MAP.....   | 50   |
| APPENDIX F: OUTPUT FILE FOR INTERACTIVE RUN.....           | 52   |
| APPENDIX G: OUTPUT FILE FOR BATCH RUN.....                 | 56   |
| APPENDIX H: CONTROL CARD FILE TO EXECUTE PROGRAM FROM      |      |
| RELOCATABLE BINARY.....                                    | 60   |
| REFERENCES.....  | 62   |
| FIGURES.....   | 63   |

## SUMMARY

Techniques for varying the segmented load structure of a program and for varying the dynamic storage allocation, depending upon whether a batch type or interactive type run is desired, are explained and demonstrated. All changes are based on a single data input to the program. The techniques involve:

1. Code within the program to suppress scratch pad input/output (IØ) for a "batch" run or translate the in-core data storage area from blank common to the end-of-code+1 address of a particular segment for an "interactive" run;
2. Automatic editing of the segload directives prior to loading, based upon data input to the program, to vary the structure of the load for "interactive" and "batch" runs; and
3. Automatic editing of the load map to determine the initial addresses for in-core data storage for an "interactive" run.

## INTRODUCTION

The purpose of this paper is to explain a successful method of incorporating the advantages of overlay into segmented loads while retaining all the dynamic load advantages of segmentation. The user is allowed to automatically load a program in a variable manner, based solely upon data input, to maintain field lengths under 100K for interactive use, which allows improved turn-a-round time and program interaction, or to maintain minimal IØ for batch runs, and thereby reduce job costs.

The overlay loader allows a program to be structured into portions, called overlays, only one of which resides in core at a time. However, this structuring has to be set up in program code, and changes in structure require changes in code and re-compilation of the program. The segment loader allows a relocatable binary program to be structured into portions, called segments, at load time. No re-coding and no re-compilation are necessary.

This paper presents the three basic steps necessary to allow automatic varying of the dynamic storage allocation and the loading structure to best suit the mode of operation desired - batch or interactive. They are:

1. Suppression of IØ for batch type runs or the translation of in-core data storage areas from blank

common to the end-of-code+1 address of a given segment for interactive runs;

2. Altering of the loading directives to vary the structure of the loads for interactive or batch type runs; and
3. Determination of the initial addresses from the load map, needed for the in-core data storage in step 1.

All these steps, accomplished by the programmer in pre-written code and small files of editing commands, are automatically performed in the job stream with no action required by the user except to change one data card.

## SECTION I: PROBLEM DESCRIPTION

The basic problem which was solved using the techniques described herein was the necessity to keep the field length of a large applications program below 100K while being run interactively for program development or for quick interim analytical results and plots, where fast turn-a-round time was desirable. The same program was also used to obtain results which required many iterations through the entire program and a large amount of execution time. For this case, actual execution costs due exclusively to excessive IØ as a result of loading different segments and data during each program iteration, became increasingly significant. At this point it became desirable to run the program in the batch mode allowing the field length to increase in order to decrease the IØ.

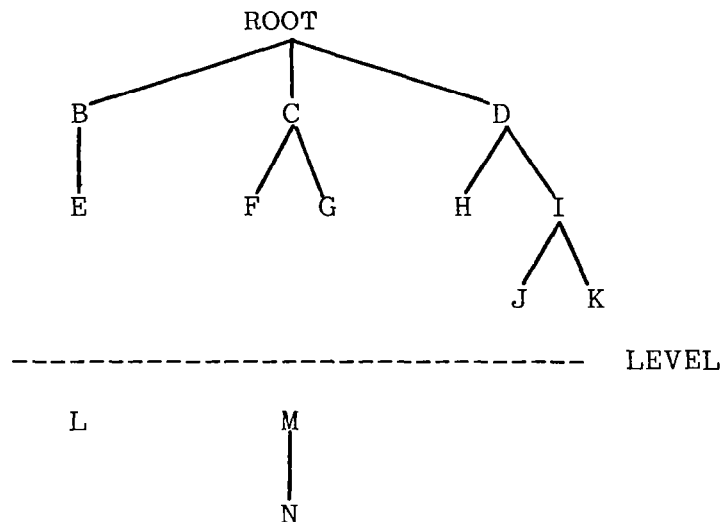
The solution to this problem is to vary the type of loading structure for interactive and batch runs, loading small segments of the program at one time during interactive runs and the entire program during batch runs. Some difficulty arises when the entire program is too large to load into core at one time in the batch mode. In this case the program must also be separated into segments, but the segments may be larger and the IØ involved in loading these segments and storing data in and out of core may be reduced.

## SECTION II: PROBLEM SOLUTION

### Load Structures

The first step in solving the problem is to determine which portions or segments of the program must be in core simultaneously. This dictates the basic tree structure for the interactive run (See ref.2 - Segmentation). A tree structure diagram is simply a diagram indicating which subroutines are to be loaded at one time and where.

ILLUSTRATION 1: Tree or Branching Diagram



The above diagram indicates that if subroutine J, for instance, were called, then all the subroutines between the ROOT and J, namely the branch containing D, I, and J, would be loaded if they were not already in core. If, say, F were then called, F and C would be loaded, over-writing D, I, and J. If L were then called, F and C would remain, untouched, since L is loaded starting at the second level. That is, subroutines residing at one level remain intact when loading subroutines into a different level.

For the batch run, cost can be effectively reduced by loading the segments or branches (combinations of segments) which are called repetitively into core simultaneously at different levels. (See ref.2 - Segmentation)

## Data Requirements for Each Branch

Once the tree structure is set up for each type of run, the data required by each segment or branch has to be determined. The data can then be stored out of core on mass storage and read into core only when its presence in core is required (See Ref.1 - Dynamic Storage Allocation). Note here, however, that unlike overlays in which blank common is loaded at the end of each overlay, requiring corresponding data to be read into core each time an overlay is loaded, in segmented loads, blank common is loaded at a level starting at the highest address of all the segments or branches which might reside in core at one time. The advantage of this when field length is not a major factor is that loading of different segments does not affect the data already in blank common. Consequently, the I/O activity associated with repetitive data reads required when incorporating dynamic storage allocation into overlay structures can be eliminated with segmentation. The disadvantage comes in interactive runs when field length is the overriding constraint for program execution.

## End-of-Code Addresses

If the objective is to reduce field length and allow as much room as possible for storage of data required by each branch (as in an interactive run), it is essential to load data immediately after the actual program code of the branch currently in core. This necessitates obtaining the actual end-of-code+1 address of each branch to be used later as an initial address for in-core data storage.

## Programming Steps

The changes in storage and access of data and the changes in the program loads are accomplished by

1. Setting up code within the program to
  - a. bypass unnecessary data reads and writes in the batch run where data is stored dynamically in core, starting with the first word of blank common, or
  - b. read data from mass storage, translating the initial address to the end-of-code+1 address in the interactive run.
2. Creating a file of editing commands to alter the segload directives from "batch" to "interactive" load directives, based upon an input parameter to



the program which the editor determines by scanning the input file.

3. Creating a file of editing commands which the editor will use to edit the load map and create an address file of initial addresses for data storage, to be input to the program during interactive runs.

### SECTION III: ILLUSTRATIONS OF TECHNIQUES

The above techniques are illustrated by exercising a sample program which is intended to simulate a large applications program. The program manipulates matrices and plots results. It does not perform any actual analyses and the results, which are not intended to be meaningful in themselves, are presented only to allow comparison between interactive and batch modes of operation.

The following illustration is an outline of the program which is to be structured at load time in two different ways. The complete program is listed in Appendix A.

#### ILLUSTRATION 2: Outline of Sample Program

```
PROGRAM MAIN(TAPE5,TAPE6,...)
COMMON /IADDRESS/ ...
COMMON /CONTROL/ ...
      :
      :
CALL DATAIN
TIME=TIMEO-DTIME
      :
      :
10  CONTINUE
    TIME=TIME+DTIME
    CALL ANALYS1
    CALL ANALYS2
    IF(TIME.LT.TEND)GO TO 10
      :
CALL PLOT
CALL DATAOUT
STOP
END

-----
SUBROUTINE DATAIN
COMMON /IADDRESS/ ...
COMMON /CONTROL/ ...
```

```

COMMON X(1)
      .
      .
CALL XINPUT
END
-----
SUBROUTINE ANALYS1
COMMON /IADDRESS/ ...
COMMON /CONTROL/ ...
COMMON X(1)
      .
CALL A
END
-----
SUBROUTINE ANALYS2
COMMON /IADDRESS/ ...
COMMON /CONTROL/ ...
COMMON X(1)
      .
CALL B
      .
CALL C
END
-----
SUBROUTINE A
      .
CALL D
      .
CALL E
      .
END
-----
SUBROUTINE B
      .
CALL E
      .
CALL F
      .
END
-----
SUBROUTINE C
      .
CALL G
      .
CALL H
      .
END
-----
SUBROUTINE PLOT
COMMON /IADDRESS/ ...
COMMON /CONTROL/ ...

```

```
COMMON X(1)
```

```
CALL PSEUDO
```

```
CALL PLOTTER
```

```
END
```

```
-----  
SUBROUTINE PSEUDO
```

(Langley graphics routine containing entry points  
CALPLT and NFRAME, used by most plot routines.  
refer to ref.3)

```
END
```

```
-----  
SUBROUTINE PLOTTER
```

```
CALL CALPLT (in PSEUDO)
```

```
CALL NOTATE
```

```
CALL BSCALE
```

```
CALL AXES
```

```
CALL LINPLT
```

```
CALL NFRAME (in PSEUDO)
```

```
END
```

```
-----  
SUBROUTINE DATAOUT
```

```
COMMON /IADDRESS/ ...
```

```
COMMON /CONTROL/ ...
```

```
COMMON X(1)
```

```
CALL XOUTPUT
```

```
END
```

### Branches and Program Code for Interactive Case

The first step as outlined in the previous section is to determine the segments or portions of the program which must reside in core simultaneously. We draw a "calling sequence" diagram, figure 1, to determine this, based upon the calling sequences in the program.

A and B both call E, hence E must reside in core whenever either A or B is in core. However, A is in the branch beginning with ANALYS1 and B is in the branch beginning with ANALYS2; therefore,

E should be loaded above ANALYS1 and ANALYS2 (See figure 2). Since PSEUDO contains entry points CALPLT and NFRAME, which are called by most plotting routines (ref.3), PSEUDO must reside in core whenever other plotting routines are in core. Furthermore, since PLOTTER calls subroutines AXES and NOTATE, and AXES also calls NOTATE (See Ref.3 - Subroutine AXES), NOTATE must be loaded whenever AXES is. Thus follows the branching diagram depicted in figure 2. Only subroutines in the same branch can reside in core at the same time. For example, G and H are in different branches and would reside in core at the same address, thus over-writing each other when loaded into core.

Now the code for the basic dynamic storage allocation (ref.1) can be inserted into the program. For this program, that means setting up initial addresses in subroutines: DATAIN, ANALYS1, ANALYS2, PLOT, and DATAOUT corresponding to each of the five primary branches. To do this, the end-of-code+1 addresses for each of these branches must have been determined and must be available for input into the program. For the sample program, these addresses will be called: IADIN, IADAN1, IADAN2, IADPLT, IADOUT. The address for the beginning of blank common will be called IADBLNK. The basic idea is to translate the beginning data storage location from blank common back to the end-of-code+1 location by giving the array word in blank common a negative index (subscript). For example, in the sample program, the array word in blank common is X(1). (Note the COMMON X(1) statement in the program.) Thus X(1) is stored at the starting address of blank common = IADBLNK. This address, established by the segment loader, is an invariant location during the entire job execution.

### ILLUSTRATION 3: Negative Indexing

```

X(1)    is at IADBLNK
X(0)    is at IADBLNK-1
X(-1)   is at IADBLNK-2
      .
      .
X(-10)  is at IADBLNK-11
      etc.

```

If IADBLNK = 75000 (octal) and the end-of-code+1 address for the current branch is 65000, then  $X(65000-75000+1) = X(-7777)$  is at the first address after the end of code. Specifically,  $X(IADIN-IADBLNK+1)$  is the first location after the end of code of the branch starting with DATAIN, and now corresponds to the address of the first piece of array data required by any of the segments in this branch. Figure 3 depicts the beginning data storage addresses for each branch. The address of the last piece of data required by each branch is computed in the program and

only the storage actually needed is called for. This may actually be less than IADBLNK for some of the branches. However, at least one branch will have an end-of-code+1 address =  $X(1) = IADBLNK$ . In the sample program, the plotting branch is the one ending just prior to blank common; i.e.,  $X(IADPLT-IADBLNK+1)=X(1)$ .

The code in the sample program (Appendix A) related to the dynamic storage allocation in the interactive case is marked by a (\*\*). This means the deviations from normal dynamic storage allocation (ref.1) are marked by a (\*\*) in program MAIN and subroutines DATAIN, ANALYS1, ANALYS2, PLOT, and DATAOUT, and, the code for reading and writing the arrays to a scratch file are marked by a (\*\*) in subroutines XINPUT, A, B, C, PLOTTER, and XOUTPUT.

#### Branches and Program Code for Batch Case

ANALYS1 and ANALYS2 might be called many times during an execution, so that for a batch run it would be desirable to load these two branches simultaneously. In this case, there are several possibilities, two of which are depicted in figures 4(a) and 4(b).

If there is sufficient field length to load and execute the program using the branching structure of figure 4(b), then this would be the best choice of the two options shown for two reasons. First, during the execution of B and C, no re-loading of segments would be necessary. Second, PSEUDO is put at the second level because normally PSEUDO and its corresponding plot routines require so much core that they would drastically increase the starting location of the second level if put into the first level, thus increasing the overall field length unnecessarily. In the following discussion and illustrations, the assumption is made that figure 4(b) is the optimal load structure for batch runs.

Earlier, it was mentioned that data in blank common is not overwritten by the segment loader when loading different segments. In the batch case, advantage can be taken of this fact in two ways. One, data used by more than one segment can be stored in the beginning of blank common. The starting address of work areas for each of these segments can then begin with the word following the last word used by these "common" arrays. (Note the beginning index = ILAST in subroutines ANALYS1 and ANALYS2, in Appendix A.) Secondly, these "common" arrays need not be stored out of core between segment calls, so the scratch-pad "writes" and "reads" can be by-passed in the batch runs. The space used by arrays which are not common to more than one branch is overwritten by successive branches. These "non-common" arrays are stored out of core only if they need to be saved for later use. Code used for the dynamic storage allocation in the batch case are marked by a (\*\*\*\*) in Appendix A.

## Load Directives

The next step is to set up the load directives for each type of load. Refer to figure 2 and figure 4(b), which depict the load structures to be used for the interactive and batch runs, respectively. Example 1 is the set of load directives for the interactive run, corresponding to figure 2, and Example 2 is the set of load directives for the batch run, corresponding to figure 4(b). Both of these sets of load directives will probably need to be modified slightly, after looking at the initial load map, with some "global" and "include" directives (see ref.2 - Segmentation) in order to globalize certain commons used by system and library routines and to adjust the locations of some of these routines. Appendix B is a listing of the final sets of load directives used for the sample program.

### EXAMPLE 1: Load Directives for Interactive Load

|      |        |                                    |
|------|--------|------------------------------------|
| ROOT | TREE   | MAIN-(IN,EE,PLT,OUT)               |
| MAIN | GLOBAL | IADDRESS,CONTROL                   |
| IN   | TREE   | DATAIN-XINPUT                      |
| EE   | TREE   | E-(AN1,AN2)                        |
| AN1  | TREE   | ANALYS1-A-D                        |
| PLT  | TREE   | PLOT-PLOTTER-PSE                   |
| OUT  | TREE   | DATAOUT-XOUTPUT                    |
| AN2  | TREE   | ANALYS2-(B-F,C-(G,H))              |
| PSE  | TREE   | PSEUDO-(BSCALE,NOTATE-AXES,LINPLT) |

### EXAMPLE 2: Load Directives for Batch Load

|       |        |                                  |
|-------|--------|----------------------------------|
| ROOT  | TREE   | MAIN-(IN,EE,PLT,OUT)             |
| MAIN  | GLOBAL | IADDRESS,CONTROL                 |
| IN    | TREE   | DATAIN-XINPUT                    |
| EE    | TREE   | E-AN1                            |
| AN1   | TREE   | ANALYS1-A-D                      |
| PLT   | TREE   | PLOT-PLOTTER                     |
| OUT   | TREE   | DATAOUT-XOUTPUT                  |
| LEVEL |        |                                  |
| AN2   | TREE   | ANALYS2-B-F-C-G-H                |
| PSE   | TREE   | PSEUDO-BSCALE-NOTATE-AXES-LINPLT |

## Editing Commands for Conversion of Load Directives

Now that the load structures and the corresponding load directives have been determined, a file of editing commands to alter the segload directives from the "batch" load to the "interactive" load (or vice versa) is required. The first data card on file TAPE5 contains either a "1" or a "0", 1 for batch, and 0 for interactive. Thus, the following command file, Example 3, will perform the conversion from "batch" to "interactive" load directives using the XEDIT editor.

### EXAMPLE 3: Editing Commands for Conversion of Load Directives from Batch Load to Interactive Load

```
IB2                (Insert 2 lines of text before present line)
*START             (Insert a "*START" at the top of load di-
                   rective file)
*END               (Insert a *END after *START statement)
^READ TAPE5        (Insert TAPE5 data in front of *END-line)
^N                 (Go to second line of file)
IF$0$              (Determine if first data card is a "0")
L$AN1$             (Locate the line containing the word "AN1")
C$-AN1$-(AN1,AN2)$ (Change character string)
L$PLOTTER$         (Locate line containing string "PLOTTER")
C$TER$TER-PSE$     (Change character string)
Z;L$LEVEL$;D       (Locate and delete line containing "LEVEL")
C$B-F-C-G-H$(B-F,C-(G,H))$1 (Change character string on next line)
L$PSEUDO$          (locate line containing "PSEUDO")
C$BSCALE-NOTATE-AXES-LINPLT$(BSCALE,NOTATE-AXES,LINPLT)$
                   (Change character string)
ELSE               (Skip to here if first data card did not
                   contain a "0")
ENDIF
TOP                (Go to top of file)
COPYD DUMMY $*END$ (Delete all lines down to *END statement)
END,LOADIR         (End editing, rename file LOADIR)
```

### Editing Commands to Create Address File

The next step is to create a file of editing commands which the editor will use to edit the load map and create an address file of initial addresses for data storage, to be input to the program during interactive runs. Note that these editing commands might have to be altered if the original tree structures are changed.

The first address to be determined is IADBLNK. Consider the following portion of the load map (Appendix C) for the interactive run.

|                     |       |
|---------------------|-------|
| FWA OF THE LOAD     | 1054  |
| LWA+1 OF THE LOAD   | 46275 |
| CM BLANK COMMON FWA | 46274 |

WRITTEN TO FILE        SAMABS

TRANSFER ADDRESS -- MAIN                    16204

Since IADBLNK is the first word of blank common, the statement

                                CM BLANK COMMON FWA  
must be located. The following commands will locate this statement, modify it to conform to data input for the program, and copy it to a file named ADRESS.

```
Z;L$ CM BLANK $;YQM
#####&IADBLNK=#&&&&&&&&                                B,#####
COPY ADRESS
```

The next address to be generated is IADIN. This is the end-of-code+1 address for the branch beginning with DATAIN. Since XINPUT is the last segment in this branch, (see figure 2.-Branching Diagram for Interactive runs or Appendix B), the portion of the load map which corresponds to the segment XINPUT must be studied.

----- SEGMENT - XINPUT

#### PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK        | ADDRESS | LENGTH | FILE | DATE     | PROCSSR |
|--------------|---------|--------|------|----------|---------|
| /IADDRESS/ S | 1061    | 25     |      |          |         |
| /CONTROL/ S  | 1106    | 22     |      |          |         |
| (XINPUT)     | 35767   | 0      |      |          |         |
| XINPUT       | 35767   | 167    | LGO  | 80/04/17 | FTN     |

XINPUT is the only subroutine in this segment, therefore, it is necessary to locate the address of XINPUT = IADIN, and its length



= IPIN, output both to file ADRESS and allow the program to add the length to the address of XINPUT to obtain the index for end-of-code+1 for the primary branch - DATAIN. The following commands will locate the line containing the proper address, modify it, and copy it to file ADRESS.

```
Z;L$ (XINPUT)$;N;YQM
#####&IADIN=&&          , IPIN= #,#####
C$,$B,$
COPY ADRESS
```

The other addresses are obtained similarly. However, in obtaining the address for the primary branch - ANALYS2, care must be exercised. Referring to figure 2 again, note that segments F, G, and H are all final segments in this branch. To be sure that no segment will overwrite the data for this branch, the address must be derived from the longest section of the branch. Consider the following portions of the load map:

----- SEGMENT - F

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK | ADDRESS | LENGTH | FILE | DATE     | PROCSSR |
|-------|---------|--------|------|----------|---------|
| (F)   | 36664   | 0      |      |          |         |
| F     | 36664   | 65     | LGO  | 80/04/17 | FTN     |

----- SEGMENT - G

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK | ADDRESS | LENGTH | FILE | DATE     | PROCSSR |
|-------|---------|--------|------|----------|---------|
| (G)   | 36620   | 0      |      |          |         |
| G     | 36620   | 72     | LGO  | 80/04/17 | FTN     |

----- SEGMENT - H

# PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK | ADDRESS | LENGTH | FILE | DATE     | PROCSSR |
|-------|---------|--------|------|----------|---------|
| (H)   | 36620   | 0      |      |          |         |
| H     | 36620   | 46     | LGO  | 80/04/17 | FTN     |

Subroutine F in SEGMENT - F has the highest address and length combination for this branch. Hence, the following commands will generate the correct values for IADAN2 and IPAN2 (address and length) to be copied to file ADDRESS.

```
Z;L$ (F) $;N;YQM
#####&IADAN2=& , IPAN2=#,#####
C$,$B,$
COPY ADDRESS
```

This same care must be taken in determining the address for the plot branch. Referring to figure 2, segments BSCALE, AXES, and LINPLT must be considered:

----- SEGMENT - BSCALE

## PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK    | ADDRESS | LENGTH | FILE       | DATE     | PROCSSR |
|----------|---------|--------|------------|----------|---------|
| (BSCALE) | 42272   | 0      |            |          |         |
| BSCALE   | 42272   | 506    | UL-LRCGOSF | 79/12/10 | FTN     |
| LEGVAR   | 43000   | 4      | SL-FORTRAN | 79/12/07 | COMPASS |
| EXP.     | 43004   | 72     | SL-FORTRAN | 79/12/07 | COMPASS |
| XTOY.    | 43076   | 31     | SL-FORTRAN | 79/12/07 | COMPASS |

----- SEGMENT - LINPLT

## PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK    | ADDRESS | LENGTH | FILE       | DATE     | PROCSSR |
|----------|---------|--------|------------|----------|---------|
| (LINPLT) | 42272   | 0      |            |          |         |
| LINPLT   | 42272   | 504    | UL-LRCGOSF | 79/12/10 | FTN     |
| CIRCLE   | 42776   | 165    | UL-LRCGOSF | 79/12/10 | FTN     |
| CNTRLN   | 43163   | 134    | UL-LRCGOSF | 79/12/10 | FTN     |
| GRID     | 43317   | 134    | UL-LRCGOSF | 79/12/10 | FTN     |
| PNTPLT   | 43453   | 524    | UL-LRCGOSF | 79/12/10 | FTN     |
| SQRT.    | 44177   | 32     | SL-FORTRAN | 79/12/07 | COMPASS |

----- SEGMENT - AXES

#### PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK       | ADDRESS | LENGTH | FILE       | DATE     | PROCSSR |
|-------------|---------|--------|------------|----------|---------|
| /FCL.C./ S  | 1130    | 26     |            |          |         |
| /Q8.IO./ S  | 1156    | 100    |            |          |         |
| /SETCHAR/ S | 37050   | 56     |            |          |         |
| (AXES)      | 44236   | 0      |            |          |         |
| /APOS/      | 44236   | 17     |            |          |         |
| AXES        | 44255   | 1524   | UL-LRCGOSF | 79/12/10 | FTN     |
| NUMBER      | 46001   | 167    | UL-LRCGOSF | 79/12/10 | FTN     |
| ENCODE=     | 46170   | 104    | SL-FORTRAN | 79/12/07 | COMPASS |

In this case, subroutine ENCODE= in SEGMENT - AXES has the highest address and length combination. The commands to generate the correct values for IADPLT and IPPLT are as follows:

```
Z;L$ ENCODE= $;YQM
#####&IADPLT&= , IPPLT=#,#####
C$,$B,$
COPY ADDRESS
```

Appendix E contains a complete listing of the command file used for editing the load map to generate addresses for the sample program. Since the addresses are not necessary for the batch

version, this editing need not be done if the first data card on file TAPE5 contains a "1". For this reason, the same bypass instructions were used as for editing the SEGLOAD directives.

#### SECTION IV: RESULTS

The use of an interactive terminal to debug and execute jobs can result in sizable savings in time and effort, but use of an interactive terminal requires that job field lengths be minimized. Methods have been described that allow automatic, variable structuring and execution of a program for interactive and batch modes of operation. It was stated that these methods could significantly reduce field lengths for interactive operation or reduce IØ and mass storage costs in batch operation. To demonstrate this, these methods were applied to the sample program as previously discussed, and both types of runs were made. The following is a comparison of the resulting field lengths and costs of the two runs.

Illustration 4 presents a comparison of the Interactive and batch field length requirements for the problem that was run. These would change with different size problems.

##### ILLUSTRATION 4: Field Lengths Used for Batch and Interactive Runs

|                          | BATCH  | INTERACTIVE |
|--------------------------|--------|-------------|
| FIELD LENGTH - DATAIN -  | 052166 | 037050      |
| FIELD LENGTH - ANALYS1 - | 053107 | 040361      |
| FIELD LENGTH - ANALYS2 - | 052546 | 040223      |
| FIELD LENGTH - PLOT -    | 052251 | 046460      |
| FIELD LENGTH - DATAOUT - | 052251 | 037253      |

The field lengths dropped 10-13K (octal), except for the plotting segment, which dropped by 4K (octal), giving an average reduction of 11K (octal) , or 22%, and a minimum reduction for the 4K of 9.5%. Larger programs can realize an even greater savings, enough usually to maintain field lengths under 100K, depending on the size of the problem, which in this case was reflected in the size of the matrices.

The CP (Central Processor) times for the batch and interactive runs were essentially the same, 19.379 seconds for the batch run and 19.758 seconds for the interactive run. The SRU (Standard Resource Unit) costs were not the same, however. The interactive run cost 94.455 SRU's as compared to only 81.174 SRU's for the batch run. This 14% reduction reflects the difference in IØ and mass storage costs of the two versions, and this reduction would increase steadily with increased number of iterations through the program. The batch version costs less to execute, but turn-a-round time is longer. The interactive version can give much more immediate results.

Identical numerical results were obtained by the batch and interactive versions of the program as can be seen by examining the plot outputs in figures 5 (a) and (b). Tabulated results are given in Appendices F and G as further evidence of correct operation of both versions.

The control deck required for job execution was the same for both batch and interactive operation. It is listed in Appendix H. All changes in restructuring and IØ are performed by prewritten code and only require the user to change a single piece of data on the first data card.

#### CONCLUDING REMARKS

The results of this sample program illustrate the dramatic savings in either field lengths or IØ costs that can be realized by variable structuring of a program. By using such techniques, large programs which normally could not be run interactively can often be sufficiently reduced in size to be executed from an interactive terminal. This is accomplished without detracting from the cost effective advantages of the batch runs. Additionally, the segment loader eliminates the necessity to re-code and re-compile in order to restructure the load to best suit each mode of operation.

## APPENDIX A

### SAMPLE PROGRAM LISTING

```

**      PROGRAM MAIN(TAPE5,TAPE6,OUTPUT,ADRESS,SCRATCH,PLTDAT)
**      COMMON /IADDRESS/ IADBLNK,IADIN,IADAN1,IADAN2,IADPLT,IADOUT
**      C          ,IPIN,IPAN1,IPAN2,IPPLT,IPOUT
**      C          ,IBATCH,ILAST,IAA,IBB,ICC,IFF
**      C          ,IXX,IY1,IY2,NPLOT
**      COMMON /CONTROL/ N1,N2,TIME0,TEND,DTIME,DTX,TIME,IPLT
**      COMMON /INDEX/ INDEX(11)
**      NAMELIST /INPUT/ N1,N2,TIME0,TEND,DTIME,DTX,IPLT

      REWIND 5
      IPLTDAT=6LPLTDAT
      REWIND IPLTDAT
      REWIND 6
**      READ(5,*)IBATCH
**      IF(IBATCH.EQ.1)GO TO 1
      CALL OPENMS(6LSCRATCH,INDEX,11,0)
**      IADDRESS=6LADDRESS
**      READ(IADDRESS,50)IADBLNK,IADIN,IPIN,IADAN1,IPAN1
**      C          ,IADAN2,IPAN2,IADPLT,IPPLT,IADOUT,IPOUT
**      WRITE(6,50)IADBLNK,IADIN,IPIN,IADAN1,IPAN1
**      C          ,IADAN2,IPAN2,IADPLT,IPPLT,IADOUT,IPOUT
**      50  FORMAT(23X,06/(13X,06,10X,06))
**      IADIN=IADIN+IPIN
**      IADAN1=IADAN1+IPAN1
**      IADAN2=IADAN2+IPAN2
**      IADPLT=IADPLT+IPPLT
**      IADOUT=IADOUT+IPOUT
**      1  CONTINUE

      READ(5,INPUT)

**      C*  SET SUFFICIENT FIELD LENGTH TO LOAD DATAIN, IF INTERACTIVE RUN
**      IF(IADIN.NE.0)CALL RFL(IADIN+100B)
      CALL DATAIN
      TIME=TIME0-DTIME
      10  CONTINUE

      TIME=TIME+DTIME

**      C*  SET SUFFICIENT FIELD LENGTH TO LOAD ANALYS1, IF INTERACTIVE
**      IF(IADAN1.NE.0)CALL RFL(IADAN1+100B)

      CALL ANALYS1

**      C*  SET SUFFICIENT FIELD LENGTH TO LOAD ANALYS2, IF INTERACTIVE
**      IF(IADAN2.NE.0)CALL RFL(IADAN2+100B)

      CALL ANALYS2

      IF(TIME.LT.TEND)GO TO 10

      IF(IPLT.NE.1)GO TO 15

```

```

** C* SET FIELD LENGTH TO LOAD PLOTTER, IF INTERACTIVE
** IF(IADPLT.NE.0)CALL RFL(IADPLT+100B)

CALL PLOT
15 CONTINUE

** C* SET FIELD LENGTH TO OUTPUT RESULTS, IF INTERACTIVE
** IF(IADOUT.NE.0)CALL RFL(IADOUT+100B)

CALL DATAOUT

STOP
END
*****
BLOCK DATA INITIAL
** COMMON /IADDRESS/ IAD(21)
COMMON /CONTROL/ ICON(8)
** COMMON /INDEX/ INDEX(11)

** DATA IAD/21*0/
C ,ICON/8*0/
** C ,INDEX/11*0/
END
*****
IDENT RFL
ENTRY RFL
RFL DATA 0
SA1 X1
BX6 X1
LX6 30
SA6 MEM
MEMORY CM, MEM, R
EQ RFL
MEM DATA 0
END
*****

```



```

      SUBROUTINE DATAIN
**      COMMON /IADDRESS/ IADBLNK,IADIN,IADAN1,IADAN2,IADPLT,IADOUT
**      C          ,IPIN,IPAN1,IPAN2,IPPLT,IPOUT
**      C          ,IBATCH,ILAST,IAA,IBB,ICC,IFF
**      C          ,IXX,IY1,IY2,NPLOT
      COMMON /CONTROL/ N1,N2,TIME0,TEND,DTIME,DTX,TIME,IPLT
      COMMON X(1)

C*  ASSUMING ARRAYS AA,BB,CC ARE TO BE INPUT AND USED BY ANALYS1
C*  AND ANALYS2, SET UP INITIAL STARTING ADDRESSES FOR DATA.

      IAA=1
**      IF(IBATCH.EQ.0)IAA=IADIN-IADBLNK+1
      IBB=IAA+N1
      ICC=IBB+N1*N2
****     ILAST=ICC+N2*N2

      IFL=ICC+N2*N2+100B+LOCF(X(1))
      CALL RFL(IFL)
      WRITE(6,50)IFL
50      FORMAT(//* FIELD LENGTH FOR BRANCH -- DATAIN -- IS *,06)
      CALL XINPUT(X(IAA),X(IBB),X(ICC),N1,N2)
      END
*****
      SUBROUTINE XINPUT(AA,BB,CC,J1,J2)
      DIMENSION AA(J1),BB(J1,J2),CC(J2,J2)
**      COMMON /IADDRESS/ IADBLNK,IADIN,IADAN1,IADAN2,IADPLT,IADOUT
**      C          ,IPIN,IPAN1,IPAN2,IPPLT,IPOUT
**      C          ,IBATCH,ILAST,IAA,IBB,ICC,IFF
**      C          ,IXX,IY1,IY2,NPLOT
      COMMON /CONTROL/ N1,N2,TIME0,TEND,DTIME,DTX,TIME,IPLT

      READ(5,*)(AA(I),I=1,N1)
      READ(5,*)((BB(I,J),J=1,N2),I=1,N1)
      READ(5,*)((CC(I,J),J=1,N2),I=1,N2)

**      C*  STORE DATA ON SCRATCH FILE IF INTERACTIVE RUN
**      IF(IBATCH.EQ.1)GO TO 10

**      CALL WRITMS(6LSCRCH,AA,N1,1,-1,0)
**      CALL WRITMS(6LSCRCH,BB,N1*N2,2,-1,0)
**      CALL WRITMS(6LSCRCH,CC,N2*N2,3,-1,0)

**      10  CONTINUE
      RETURN
      END
*****

```

```

SUBROUTINE ANALYS1
**      COMMON /IADRESS/ IADBLNK,IADIN,IADAN1,IADAN2,IADPLT,IADOUT
**      C              ,IPIN,IPAN1,IPAN2,IPPLT,IPOUT
**      C              ,IBATCH,ILAST,IAA,IBB,ICC,IFF
**      C              ,IXX,IY1,IY2,NPLOT
      COMMON /CONTROL/ N1,N2,TIME0,TEND,DTIME,DTX,TIME,IPILOT
      COMMON X(1)
C* ASSUMING INPUT ARRAYS AA AND BB, AND ARRAYS DD,EE,FF,AND PIVOT
C* ARE TO BE USED BY SUBROUTINE A,D,AND E,SET UP INITIAL ADDRESSES
C* STARTING WITH FF, BECAUSE FF IS TO BE GENERATED FOR USE BY ANALYS2

****      IF(TIME.EQ.TIME0)IFF=ILAST
****      IF(IBATCH.EQ.1)GO TO 10
**      IAA=IADAN1-IADBLNK+1
      IBB=IAA+N1
      IFF=IBB+N1*N2

**** 10      CONTINUE
****      IF(TIME.EQ.TIME0)ILAST=IFF+N1*N1
      IDD=IFF+N1*N1
      IEE=IDD+N2**2
      IPIV=IEE+N1*N1

      IFL=IPIV+4*N1+100B+LOCF(X(1))
      CALL RFL(IFL)
      IF(TIME.EQ.TIME0)WRITE(6,50)IFL
50      FORMAT(//* FIELD LENGTH FOR BRANCH -- ANALYS1 -- IS *,O6)

      CALL A(X(IAA),X(IBB),X(IDD),X(IEE),X(IFF),X(IPIV),N1,N2)
      RETURN
      END
*****
      SUBROUTINE A(AA,BB,DD,EE,FF,IPIV,J1,J2)
      DIMENSION AA(J1),BB(J1,J2),DD(J2,J2),EE(J1,J1),FF(J1,J1)
      DIMENSION IPIV(1)
**      COMMON /IADRESS/ IADBLNK,IADIN,IADAN1,IADAN2,IADPLT,IADOUT
**      C              ,IPIN,IPAN1,IPAN2,IPPLT,IPOUT
**      C              ,IBATCH,ILAST,IAA,IBB,ICC,IFF
**      C              ,IXX,IY1,IY2,NPLOT
      COMMON /CONTROL/ N1,N2,TIME0,TEND,DTIME,DTX,TIME,IPILOT

**      C* READ ARRAYS AA AND BB INTO CORE IF INTERACTIVE RUN
**      IF(IBATCH.EQ.1)GO TO 10

**      CALL READMS(6LSCRTCH,AA,N1,1)
**      CALL READMS(6LSCRTCH,BB,N1*N2,2)

**      10      CONTINUE

C* CALL SUBROUTINE D TO GENERATE
C*      DD=(BB-AA*I*TIME)'*(BB-AA*I*TIME) ('=TRANPOSE)

```

```

C*          EE=AA*AA'
C*          FF=BB*DD*BB'- EE

      CALL D(AA,BB,DD,EE,FF,N1,N2,TIME)

C*  CALL SUBROUTINE E TO COMPUTE FF-INVERSE & DET(FF)

      CALL E(FF,N1,DUMMY,0,KEY,DET,IPIV,IPIV(1+N1),IPIV(1+3*N1))

      IPLTDAT=6LPLTDAT
      WRITE(IPLTDAT)TIME,DET

**  C*  STORE FF-INVERSE ON SCRATCH IF INTERACTIVE RUN
      IF(IBATCH.EQ.0)CALL WRITMS(6LSCRTCH,FF,N1*N1,4,-1,0)

      RETURN
      END
*****
      SUBROUTINE D(AA,BB,DD,EE,FF,I1,I2,TIME)
      DIMENSION AA(I1),BB(I1,I2),DD(I2,I2),EE(I1,I2),FF(I1,I1)

C*  COMPUTE DD=(BB-I*AA*TIME)'(BB-I*AA*TIME)

      DO 2 I=1,I1
      DO 1 J=1,I2
1      EE(I,J)=BB(I,J)
2      EE(I,I)=BB(I,I)-AA(I)*TIME

      DO 3 J=1,I2
      DO 3 I=1,I2
      DD(I,J)=0.
      DO 3 K=1,I1
3      DD(I,J)=DD(I,J)+EE(K,I)*EE(K,J)

C*  COMPUTE EE=AA*AA'

      DO 10 J=1,I2
      DO 10 I=1,I1
10      EE(I,J)=AA(I)*AA(J)

C*  COMPUTE FF=BB*DD*BB'- EE

      DO 20 J=1,I1
      DO 20 I=1,I1
      FF(I,J)=-EE(I,J)
      DO 20 K=1,I2
      DO 20 L=1,I2
20      FF(I,J)=FF(I,J)+BB(I,K)*DD(K,L)*BB(J,L)

      RETURN

```

```

      END
*****
      SUBROUTINE E(A,N,B,M,KEY,DETERM,IPIVOT,INDEX,PIVOT)
      DIMENSION A(N,1),B(1),IPIVOT(1),INDEX(N,2),PIVOT(1)
C      INITIALIZATION
        5 KEY = N
        10 DETERM=1.
        15 DO 20 J=1,N
        20 IPIVOT(J)=0
        30 DO 550 I=1,N
C
C      SEARCH FOR PIVOT ELEMENT
        40 AMAX=0.0
        45 DO 105 J=1,N
        50 IF (IPIVOT(J)-1) 60,105,60
        60 DO 100 K=1,N
        70 IF (IPIVOT(K)-1) 80,100,740
        80 IF(ABS(AMAX)-ABS(A(J,K))) 85,100,100
        85 IROW=J
        90 ICOLUM=K
        95 AMAX=A(J,K)
        100 CONTINUE
        105 CONTINUE
        ZERO=1.E-16
        107 IF(ABS(AMAX)-ZERO)745,745,110
        110 IPIVOT(ICOLUM)=IPIVOT(ICOLUM)+1
C
C      INTERCHANGE ROWS TO PUT PIVOT ELEMENT ON DIAGONAL
C
        130 IF (IROW-ICOLUM) 140,260,140
        140 DETERM=-DETERM

        150 DO 200 L=1,N
        160 SWAP=A(IROW,L)
        170 A(IROW,L)=A(ICOLUM,L)
        200 A(ICOLUM,L)=SWAP
        205 IF(M) 260,260,210
        210 SWAP = B(IROW)
        230 B(IROW) = B(ICOLUM)
        250 B(ICOLUM) = SWAP
        260 INDEX(I,1)=IROW
        270 INDEX(I,2)=ICOLUM
        310 PIVOT(I)=A(ICOLUM,ICOLUM)
C
C      DIVIDE PIVOT ROW BY PIVOT ELEMENT
C
        330 A(ICOLUM,ICOLUM)=.1E+1
        340 DO 350 L=1,N
        350 A(ICOLUM,L)=A(ICOLUM,L)/PIVOT(I)
        355 IF(M) 380,380,360
        360 B(ICOLUM) = B(ICOLUM)/PIVOT(I)

```

```

C
C      REDUCE NON-PIVOT ROWS
C
380 DO 550 L1=1,N
390 IF(L1-ICOLUM) 400,550,400
400 T=A(L1,ICOLUM)
420 A(L1,ICOLUM)=0.0
430 DO 450 L=1,N
450 A(L1,L)=A(L1,L)-A(ICOLUM,L)*T
455 IF(M) 550,550,460
460 B(L1) = B(L1) - B(ICOLUM)*T
550 CONTINUE
C
C      INTERCHANGE COLUMNS
C
600 DO 710 I=1,N
610 L=N+1-I
620 IF (INDEX(L,1)-INDEX(L,2)) 630,710,630
630 JROW=INDEX(L,1)
640 JCOLUM=INDEX(L,2)
650 DO 705 K=1,N
660 SWAP=A(K,JROW)
670 A(K,JROW)=A(K,JCOLUM)
700 A(K,JCOLUM)=SWAP
705 CONTINUE
710 CONTINUE
      DO 800 I=1,N
      J=N+1-I
800 DETERM=DETERM*PIVOT(J)

740 RETURN
745 DETERM=0.0
746 KEY = I - 1
750 RETURN
      END

```

\*\*\*\*\*

```

      SUBROUTINE ANALYS2
**      COMMON /IADDRESS/ IADBLNK,IADIN,IADAN1,IADAN2,IADPLT,IADOUT
**      C          ,IPIN,IPAN1,IPAN2,IPPLT,IPOUT
**      C          ,IBATCH,ILAST,IAA,IBB,ICC,IFF
**      C          ,IXX,IY1,IY2,NPLOT
      COMMON /CONTROL/ N1,N2,TIME0,TEND,DTIME,DTX,TIME,IPILOT
      COMMON X(1)

C*  ASSUMING "COMMON" ARRAYS AA,BB,CC,FF,AND ARRAYS FFB,GG,AND
C*  PIVOT ARE TO BE USED BY B AND C, SET UP INITIAL ADDRESSES

****      IFFB=ILAST
****      IF(IBATCH.EQ.1)GO TO 10

**      IAA=IADAN2-IADBLNK+1
      IBB=IAA+N1
      ICC=IBB+N1*N2
      IFF=ICC+N2*N2
      IFFB=IFF+N1*N1

**** 10      CONTINUE
      IPIV=IFFB+N1*N1
      IFL=IPIV+4*N1+LOC(X(1))+100B
      CALL RFL(IFL)
      IF(TIME.EQ.TIME0)WRITE(6,50)IFL
50      FORMAT(//* FIELD LENGTH FOR BRANCH -- ANALYS2 -- IS *,06)

      CALL B(X(IAA),X(IBB),X(IFF),X(IFFB),X(IPIV),N1,N2)

      IGG=IFFB
      CALL C(X(IFF),X(IBB),X(ICC),X(IGG),N1,N2)

      RETURN
      END
*****
      SUBROUTINE B(AA,BB,FF,FFB,IPIV,J1,J2)
      DIMENSION BB(J1,J2),AA(J1),FF(J1,J1),FFB(J1,J1),IPIV(1)
**      COMMON /IADDRESS/ IADBLNK,IADIN,IADAN1,IADAN2,IADPLT,IADOUT
**      C          ,IPIN,IPAN1,IPAN2,IPPLT,IPOUT
**      C          ,IBATCH,ILAST,IAA,IBB,ICC,IFF
**      C          ,IXX,IY1,IY2,NPLOT
      COMMON /CONTROL/ N1,N2,TIME0,TEND,DTIME,DTX,TIME,IPILOT

**  C*  READ ARRAYS AA,BB,AND,FF,INTO CORE IF INTERACTIVE RUN
**      IF(IBATCH.EQ.1)GO TO 10

**      CALL READMS(6LSCRTCH,AA,N1,1)
**      CALL READMS(6LSCRTCH,BB,N1*N2,2)
**      CALL READMS(6LSCRTCH,FF,N1*N1,4)

**  10      CONTINUE
C*  COMPUTE FFB=FF*BB*BB'

```

```

      CALL F(FF,BB,FFB,N1,N2)

C*  SOLVE FFB*XX=AA
      DO 20 I=1,N1
20   FFB(I,1)=AA(I)

      CALL E(FFB,N1,FF,1,KEY,DET,IPIV,IPIV(1+N1),IPIV(1+3*N1))

      RETURN
      END
*****
      SUBROUTINE F(X,Y,Z,I1,I2)
      DIMENSION X(I1,I1),Y(I1,I2),Z(I1,I1)

C*  COMPUTE Z=X*Y*Y'

      DO 1 J=1,I1
      DO 1 I=1,I1
      Z(I,J)=0.
      DO 1 K=1,I1
      DO 1 L=1,I2
1     Z(I,J)=Z(I,J)+X(I,K)*Y(K,L)*Y(J,L)

      RETURN
      END
*****
      SUBROUTINE C(FF,BB,CC,GG,J1,J2)
      DIMENSION FF(J1),BB(J1,J2),CC(J2,J2),GG(J1,J1)
**      COMMON /IADDRESS/ IADBLNK,IADIN,IADAN1,IADAN2,IADPLT,IADOUT
**      C          ,IPIN,IPAN1,IPAN2,IPPLT,IPOUT
**      C          ,IBATCH,ILAST,IAA,IBB,ICC,IFF
**      C          ,IXX,IY1,IY2,NPLOT
      COMMON /CONTROL/ N1,N2,TIME0,TEND,DTIME,DTX,TIME,IPLT
**
      IF(IBATCH.EQ.0)CALL READMS(6LSCRTCH,CC,N2*N2,3)

C*  COMPUTE GG=BB*CC*BB'
      CALL G(BB,CC,GG,N1,N2)

C*  COMPUTE FOFT=(GG*FF)'*(GG*FF)
      CALL H(FF,GG,FOFT,N1)

      IPLTDAT=6LPLTDAT
      WRITE(IPLTDAT) FOFT
      RETURN
      END
*****
      SUBROUTINE G(X,Y,Z,I1,I2)
      DIMENSION X(I1,I2),Y(I2,I2),Z(I1,I1)

C*  COMPUTE Z=X*Y*X'
      DO 10 J=1,I1

```

```

      DO 10 I=1,I1
      Z(I,J)=0.
      DO 10 K=1,I2
      DO 10 L=1,I2
10    Z(I,J)=Z(I,J)+X(I,K)*Y(K,L)*X(J,L)

```

```

      RETURN

```

```

      END

```

```

*****

```

```

      SUBROUTINE H(X,Y,FOFT,I1)

```

```

      DIMENSION X(I1),Y(I1,I1)

```

```

      FOFT=0.

```

```

      DO 10 J=1,I1

```

```

      SUM=0.

```

```

      DO 20 K=1,I1

```

```

20    SUM=SUM+X(K)*Y(K,J)

```

```

10    FOFT=FOFT+SUM*SUM

```

```

      RETURN

```

```

      END

```

```

*****

```



```

SUBROUTINE PLOT
**      COMMON /IADRESS/ IADBLNK,IADIN,IADAN1,IADAN2,IADPLT,IADOUT
**      C          ,IPIN,IPAN1,IPAN2,IPPLT,IPOUT
**      C          ,IBATCH,ILAST,IAA,IBB,ICC,IFF
**      C          ,IXX,IY1,IY2,NPLOT
      COMMON /CONTROL/ N1,N2,TIME0,TEND,DTIME,DTX,TIME,IPLT
      COMMON X(1)

C*  COMPUTE NUMBER OF PLOT POINTS
      NPLOT=(TEND-TIME0)/DTIME+1

**** C*  SINCE THE FF ARRAY IS NO LONGER NEEDED, PLOT ARRAYS
**** C*  CAN BE LOCATED STARTING IN FF LOCATION FOR BATCH RUNS

****      IXX=IFF

**      IF(IBATCH.EQ.0) IXX=IADPLT-IADBLNK+1
      IY1=IXX+NPLOT+2
      IY2=IY1+NPLOT+2
      IFL=IY2+NPLOT+2+LOC(X(1))+100B
      CALL RFL(IFL)
      WRITE(6,50) IFL
50      FORMAT(// * FIELD LENGTH FOR BRANCH -- PLOT -- IS *,06)
      CALL PSEUDO
      CALL PLOTTER(X(IXX),X(IY1),X(IY2))
      RETURN
      END

*****
      SUBROUTINE PLOTTER(XX,Y1,Y2)
**      COMMON /IADRESS/ IADBLNK,IADIN,IADAN1,IADAN2,IADPLT,IADOUT
**      C          ,IPIN,IPAN1,IPAN2,IPPLT,IPOUT
**      C          ,IBATCH,ILAST,IAA,IBB,ICC,IFF
**      C          ,IXX,IY1,IY2,NPLOT
      COMMON /CONTROL/ N1,N2,TIME0,TEND,DTIME,DTX,TIME,IPLT
      DIMENSION XX(1),Y1(1),Y2(1)
      DIMENSION TITLE(5)
      DATA TITLE/10H      PLOTS ,10HVS. TIME F,
C10HOR SAMPLE ,10HPROGRAM - /

      TITLE(5)=10HINTERACTIV
      IF(IBATCH.EQ.1) TITLE(5)=10HBATCH
      CALL NOTATE(.5,.5,.2,TITLE,0.,50)
      CALL CALPLT(1.5,1.5,-3)
      IPLTDAT=6LPLTDAT
      REWIND IPLTDAT

      DO 10 I=1,NPLOT
      READ(IPLTDAT)XX(I),Y1(I)
10      READ(IPLTDAT)Y2(I)
**      C*  WRITE ALL PLOT DATA OUT TO SCRATCH FILE FOR USE IN XOUPUT
**      C*  IF INTERACTIVE RUN
**      IF(IBATCH.EQ.0) CALL WRITMS(6LSCRATCH,XX,3*(NPLOT+2),5,-1,0)

```

```

CALL BSCALE(Y1,4.,NPLOT,1,1.,0,0.)
CALL BSCALE(Y2,4.,NPLOT,1,1.,-1,0.)

DIFF=TEND-TIME0
SCF=DIFF/6.
XX(NPLOT+1)=TIME0
XX(NPLOT+2)=SCF
IF(DTX.EQ.0.)DTX=DTIME
TMAJ=6.*DTX/DIFF
TMIN=2./TMAJ
CALL AXES(0.,0.,0.,6.,TIME0,SCF,TMAJ,TMIN,4HTIME,.2,-4)
CALL AXES(0.,0.,90.,4.,Y1(NPLOT+1),Y1(NPLOT+2),1.,2.,3HDET,.2,3)
CALL LINPLT(XX,Y1,NPLOT,1,1,22,1,0)
CALL CALPLT(0.,5.,-3)
CALL AXES(0.,0.,0.,6.,TIME0,SCF,TMAJ,TMIN,4HTIME,.2,-4)
CALL AXES(0.,0.,90.,4.,Y2(NPLOT+1),Y2(NPLOT+2),1.,2.,
C 4HF(T),.2,4)
CALL LINPLT(XX,Y2,NPLOT,1,1,22,1,0)
CALL NFRAME

RETURN
END

```

\*\*\*\*\*

```

SUBROUTINE DATAOUT
**      COMMON /IADRESS/ IADBLNK,IADIN,IADAN1,IADAN2,IADPLT,IADOUT
**      C              ,IPIN,IPAN1,IPAN2,IPPLT,IPOUT
**      C              ,IBATCH,ILAST,IAA,IBB,ICC,IFF
**      C              ,IXX,IY1,IY2,NPLOT
      COMMON /CONTROL/ N1,N2,TIME0,TEND,DTIME,DTX,TIME,IPLT
      COMMON X(1)

C*  OUTPUT INPUT DATA AND PLOT DATA

****      IF(IBATCH.EQ.1)GO TO 10
**      IAA=IADOUT-IADBLNK+1
      IBB=IAA+N1
      ICC=IBB+N1*N2
      IXX=ICC+N2*N2
      IY1=IXX+NPLOT+2
      IY2=IY1+NPLOT+2

**** 10      CONTINUE
      IFL=IY2+NPLOT+2+LOC(X(1))+100B
      CALL RFL(IFL)
      WRITE(6,50)IFL
50      FORMAT(/* FIELD LENGTH FOR BRANCH -- DATAOUT -- IS *,06)
      CALL XOUTPUT(X(IAA),X(IBB),X(ICC),X(IXX),X(IY1),X(IY2),N1,N2)
      RETURN
      END

*****
      SUBROUTINE XOUTPUT(AA,BB,CC,XX,Y1,Y2,J1,J2)
      DIMENSION AA(J1),BB(J1,J2),CC(J2,J2),XX(1),Y1(1),Y2(1)
**      COMMON /IADRESS/ IADBLNK,IADIN,IADAN1,IADAN2,IADPLT,IADOUT
**      C              ,IPIN,IPAN1,IPAN2,IPPLT,IPOUT
**      C              ,IBATCH,ILAST,IAA,IBB,ICC,IFF
**      C              ,IXX,IY1,IY2,NPLOT
      COMMON /CONTROL/ N1,N2,TIME0,TEND,DTIME,DTX,TIME,IPLT

C*  READ IN ARRAYS AA,BB,CC,XX,Y1,AND Y2 IF INTERACTIVE RUN

**      IF(IBATCH.EQ.1)GO TO 10

**      CALL READMS(6LSCRTCH,AA,N1,1)
**      CALL READMS(6LSCRTCH,BB,N1*N2,2)
**      CALL READMS(6LSCRTCH,CC,N2*N2,3)
**      CALL READMS(6LSCRTCH,XX,3*(NPLOT+2),5)
**
** 10      CONTINUE

C*  WRITE OUTPUT

      WRITE(6,100)(AA(I),I=1,N1)
100      FORMAT(/* INITIAL AA ARRAY*/(5G15.7))
      WRITE(6,200)N1,N2
200      FORMAT(/* INITIAL BB(*I3*,*I3*) ARRAY*)

```

```

      DO 20 I=1,N1
20    WRITE(6,201)(BB(I,J),J=1,N2)
201  FORMAT(5G15.7)
      WRITE(6,300)N2,N2
300  FORMAT(//* INITIAL CC(*I3*,*I3*) ARRAY*)
      DO 30 I=1,N2
30    WRITE(6,201)(CC(I,J),J=1,N2)
      WRITE(6,400)
400  FORMAT(T10*TIME*T20*DET*T35*F(T)*/)
      DO 40 I=1,NPLOT
40    WRITE(6,401)XX(I),Y1(I),Y2(I)
401  FORMAT(3G15.4)

      RETURN
      END

```

## APPENDIX B

### SEGLOAD DIRECTIVES AND TREE DIAGRAMS FOR BATCH AND INTERACTIVE RUNS

# SEGLOAD DIRECTIVES. (For Interactive Runs)

|        |        |                                    |
|--------|--------|------------------------------------|
| ROOT   | TREE   | MAIN-(IN,EE,PLT,OUT)               |
| MAIN   | GLOBAL | IADDRESS,CONTROL                   |
| MAIN   | GLOBAL | FCL.C.,Q8.IO.,FCL=ENT,IO.BUF.      |
| IN     | TREE   | DATAIN-XINPUT                      |
| EE     | TREE   | E-(AN1,AN2)                        |
| AN1    | TREE   | ANALYS1-A-D                        |
| PLT    | TREE   | PLOT-PLOTTER-PSE                   |
| OUT    | TREE   | DATAOUT-XOUTPUT                    |
| AN2    | TREE   | ANALYS2-(B-F,C-(G,H))              |
| PSE    | TREE   | PSEUDO-(BSCALE,NOTATE-AXES,LINPLT) |
| PSEUDO | GLOBAL | SETCHAR,LANGLEY                    |
|        | END    |                                    |

## TREE DIAGRAM. (For Interactive Runs)

```

*MAIN
?
? _DATAIN
? ?
? ? _XINPUT
?
? _E
? ?
? ? _ANALYS1
? ? ?
? ? ? _A
? ? ?
? ? ? _D
? ?
? ? _ANALYS2
? ?
? ? _B
? ? ?
? ? ? _F
? ?
? ? _C
? ?
? ? _G
? ?
? ? _H
?
? _PLOT
? ?
? ? _PLOTTER
? ?
? ? _PSEUDO
? ?

```

```

?      ?_BSCALE
?      ?
?      ?_NOTATE
?      ? ?
?      ? ?_AXES
?      ?
?      ?_LINPLT
?
?_DATAOUT
?
?_XOUTPUT

```

SEGLOAD DIRECTIVES. (For Batch Runs)

```

ROOT    TREE      MAIN-(IN,EE,PLT,OUT)
MAIN    GLOBAL    IADDRESS,CONTROL
MAIN    GLOBAL    FCL.C.,Q8.IO.,FCL=ENT,IO.BUF.
IN       TREE      DATAIN-XINPUT
EE       TREE      E-AN1
AN1      TREE      ANALYS1-A-D
PLT      TREE      PLOT-PLOTTER
PLOT     INCLUDE   PLOT,NOTATE
OUT      TREE      DATAOUT-XOUTPUT
        LEVEL
AN2      TREE      ANALYS2-B-F-C-G-H
PSE      TREE      PSEUDO
PSEUDO   INCLUDE   PSEUDO,AXES,CHARACT,CHNGSET,WHERE
PSEUDO   GLOBAL    SETCHAR,LANGLEY
        END

```

TREE DIAGRAM. (For Batch Runs)

```

*MAIN
?
?_DATAIN
? ?
? ?_XINPUT
?
?_E
? ?
? ?_ANALYS1
? ?
? ?_A
? ?
? ?_D
?
?_PLOT
? ?
? ?_PLOTTER
?
?_DATAOUT

```

?  
?\_XOUTPUT

---

\*ANALYS2

?  
?\_B  
?  
?\_F  
?  
?\_C  
?  
?\_G  
?  
?\_H

\*PSEUDO

?  
?\_BSCALE  
?  
?\_NOTATE  
?  
?\_AXES  
?  
?\_LINPLT



## APPENDIX C

LOAD MAP FOR INTERACTIVE RUN

FWA OF THE LOAD           1054  
LWA+1 OF THE LOAD       46275  
CM BLANK COMMON FWA     46274

WRITTEN TO FILE       SAMABS

TRANSFER ADDRESS -- MAIN           16204

----- SEGMENT - MAIN

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK        | ADDRESS | LENGTH | FILE | DATE | PROCSSR |
|--------------|---------|--------|------|------|---------|
| (MAIN)       | 1054    | 5      |      |      |         |
| /IADDRESS/ G | 1061    | 25     |      |      |         |

| BLOCK       | ADDRESS | LENGTH | FILE       | DATE     | PROCSSR |
|-------------|---------|--------|------------|----------|---------|
| /CONTROL/ G | 1106    | 22     |            |          |         |
| /FCL.C./ G  | 1130    | 26     |            |          |         |
| /Q8.IO./ G  | 1156    | 100    |            |          |         |
| /FCL=ENT/ G | 1256    | 40     |            |          |         |
| /IO.BUF./ G | 1316    | 227    |            |          |         |
| /INDEX/     | 1545    | 13     |            |          |         |
| /STP.END/   | 1560    | 1      |            |          |         |
| MAIN        | 1561    | 14645  | LGO        | 80/04/17 | FTN     |
| INITIAL     | 16426   | 0      | LGO        | 80/04/17 | FTN     |
| RFL         | 16426   | 10     | LGO        | 80/04/17 | COMPASS |
| Q2NTRY=     | 16436   | 0      | SL-FORTRAN | 79/12/07 | COMPASS |
| COMIO=      | 16436   | 33     | SL-FORTRAN | 79/12/07 | COMPASS |
| FCL=FDL     | 16471   | 40     | SL-FORTRAN | 79/12/07 | COMPASS |
| FECMSK=     | 16531   | 41     | SL-FORTRAN | 79/12/07 | COMPASS |
| FEIFST=     | 16572   | 3      | SL-FORTRAN | 79/12/07 | COMPASS |
| FLTIN=      | 16575   | 156    | SL-FORTRAN | 79/12/07 | COMPASS |
| FLTOUT=     | 16753   | 311    | SL-FORTRAN | 79/12/07 | COMPASS |
| FMTAP=      | 17264   | 357    | SL-FORTRAN | 79/12/07 | COMPASS |
| FORSYS=     | 17643   | 300    | SL-FORTRAN | 79/12/07 | COMPASS |
| FORUTL=     | 20143   | 46     | SL-FORTRAN | 79/12/07 | COMPASS |
| GETFIT=     | 20211   | 57     | SL-FORTRAN | 79/12/07 | COMPASS |
| INCOM=      | 20270   | 145    | SL-FORTRAN | 79/12/07 | COMPASS |
| INPC=       | 20435   | 207    | SL-FORTRAN | 79/12/07 | COMPASS |
| INPF=       | 20644   | 241    | SL-FORTRAN | 79/12/07 | COMPASS |
| KODER=      | 21105   | 451    | SL-FORTRAN | 79/12/07 | COMPASS |
| KRAKER=     | 21556   | 375    | SL-FORTRAN | 79/12/07 | COMPASS |
| LDIN=       | 22153   | 260    | SL-FORTRAN | 79/12/07 | COMPASS |

|          |       |      |            |          |         |
|----------|-------|------|------------|----------|---------|
| NAMIN=   | 22433 | 523  | SL-FORTRAN | 79/12/07 | COMPASS |
| OPENMS   | 23156 | 13   | SL-FORTRAN | 79/12/07 | COMPASS |
| OUTC=    | 23171 | 150  | SL-FORTRAN | 79/12/07 | COMPASS |
| OUTCOM=  | 23341 | 154  | SL-FORTRAN | 79/12/07 | COMPASS |
| READMS   | 23515 | 13   | SL-FORTRAN | 79/12/07 | COMPASS |
| REWIND=  | 23530 | 33   | SL-FORTRAN | 79/12/07 | COMPASS |
| SPA=     | 23563 | 11   | SL-FORTRAN | 79/12/07 | COMPASS |
| STLCRM=  | 23574 | 21   | SL-FORTRAN | 79/12/07 | COMPASS |
| STLIBI.  | 23615 | 24   | SL-FORTRAN | 79/12/07 | COMPASS |
| STLICO.  | 23641 | 16   | SL-FORTRAN | 79/12/07 | COMPASS |
| STLOBI.  | 23657 | 15   | SL-FORTRAN | 79/12/07 | COMPASS |
| STLOCO.  | 23674 | 12   | SL-FORTRAN | 79/12/07 | COMPASS |
| STLREW.  | 23706 | 14   | SL-FORTRAN | 79/12/07 | COMPASS |
| WRITMS   | 23722 | 13   | SL-FORTRAN | 79/12/07 | COMPASS |
| SYSaid=  | 23735 | 1    | SL-FORTRAN | 79/12/07 | COMPASS |
| RANCAP=  | 23736 | 410  | SL-FORTRAN |          |         |
| CPU.SYS  | 24346 | 40   | SL-SYSLIB  | 79/06/05 | COMPASS |
| CTL\$PTL | 24406 | 34   | SL-SYSLIB  | 80/01/09 | COMPASS |
| CTL\$RM  | 24442 | 434  | SL-SYSLIB  | 80/01/09 | COMPASS |
| CTL\$WR  | 25076 | 40   | SL-SYSLIB  | 80/01/09 | COMPASS |
| RBL\$RM  | 25136 | 0    | SL-SYSLIB  | 80/01/09 | COMPASS |
| CPUCPM   | 25136 | 5    | SL-SYSLIB  | 80/01/17 | COMPASS |
| ERR\$RM  | 25143 | 25   | SL-SYSLIB  | 80/01/09 | COMPASS |
| LIST\$RM | 25170 | 67   | SL-SYSLIB  | 80/01/09 | COMPASS |
| ERRCAP=  | 25257 | 321  | SL-FORTRAN |          |         |
| FERCAP=  | 25600 | 202  | SL-FORTRAN |          |         |
| FCL\$RM  | 26002 | 42   | SL-BAMLIB  | 80/01/09 | COMPASS |
| PLG\$RM  | 26044 | 0    | SL-BAMLIB  | 80/01/09 | COMPASS |
| CLSF\$RM | 26044 | 430  | SL-BAMLIB  |          |         |
| DF\$CRM  | 26474 | 226  | SL-BAMLIB  |          |         |
| EF\$CRM  | 26722 | 446  | SL-BAMLIB  |          |         |
| GET\$SQ  | 27370 | 574  | SL-BAMLIB  |          |         |
| GET\$W   | 30164 | 50   | SL-BAMLIB  |          |         |
| GET\$Z   | 30234 | 130  | SL-BAMLIB  |          |         |
| GET\$WA  | 30364 | 173  | SL-BAMLIB  |          |         |
| GPTM\$SQ | 30557 | 424  | SL-BAMLIB  |          |         |
| OPEN\$RM | 31203 | 1324 | SL-BAMLIB  |          |         |
| PUT\$SQ  | 32527 | 532  | SL-BAMLIB  |          |         |
| PUT\$CI  | 33261 | 65   | SL-BAMLIB  |          |         |
| PUT\$W   | 33346 | 215  | SL-BAMLIB  |          |         |
| PUT\$Z   | 33563 | 64   | SL-BAMLIB  |          |         |
| PUT\$WA  | 33647 | 170  | SL-BAMLIB  |          |         |
| REW\$SQ  | 34037 | 125  | SL-BAMLIB  |          |         |
| SKFL\$SQ | 34164 | 36   | SL-BAMLIB  |          |         |
| WEOX\$SQ | 34222 | 137  | SL-BAMLIB  |          |         |
| CLSV\$SQ | 34361 | 220  | SL-BAMLIB  |          |         |
| COMM\$WA | 34601 | 306  | SL-BAMLIB  |          |         |
| GET\$FU  | 35107 | 113  | SL-BAMLIB  |          |         |
| LBUF\$RM | 35222 | 134  | SL-BAMLIB  |          |         |

|         |       |     |           |
|---------|-------|-----|-----------|
| RPE\$SQ | 35356 | 155 | SL-BAMLIB |
| WAR\$SQ | 35533 | 160 | SL-BAMLIB |

----- SEGMENT - DATAIN

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK        | ADDRESS | LENGTH | FILE | DATE     | PROCSSR |
|--------------|---------|--------|------|----------|---------|
| /IADDRESS/ S | 1061    | 25     |      |          |         |
| /CONTROL/ S  | 1106    | 22     |      |          |         |
| (DATAIN)     | 35713   | 1      |      |          |         |
| DATAIN       | 35714   | 53     | LGO  | 80/04/17 | FTN     |

----- SEGMENT - XINPUT

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK        | ADDRESS | LENGTH | FILE | DATE     | PROCSSR |
|--------------|---------|--------|------|----------|---------|
| /IADDRESS/ S | 1061    | 25     |      |          |         |
| /CONTROL/ S  | 1106    | 22     |      |          |         |
| (XINPUT)     | 35767   | 0      |      |          |         |
| XINPUT       | 35767   | 167    | LGO  | 80/04/17 | FTN     |

----- SEGMENT - E

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK       | ADDRESS | LENGTH | FILE | DATE     | PROCSSR |
|-------------|---------|--------|------|----------|---------|
| /IO.BUF./ S | 1316    | 227    |      |          |         |
| (E)         | 35713   | 0      |      |          |         |
| BLOCK       | ADDRESS | LENGTH | FILE | DATE     | PROCSSR |
| E           | 35713   | 321    | LGO  | 80/04/17 | FTN     |

OUTB= 36234 174 SL-FORTRAN 79/12/07 COMPASS

----- SEGMENT - ANALYS1

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK        | ADDRESS | LENGTH | FILE | DATE     | PROCSSR |
|--------------|---------|--------|------|----------|---------|
| /IADDRESS/ S | 1061    | 25     |      |          |         |
| /CONTROL/ S  | 1106    | 22     |      |          |         |
| (ANALYS1)    | 36430   | 1      |      |          |         |
| ANALYS1      | 36431   | 105    | LGO  | 80/04/17 | FTN     |

----- SEGMENT - ANALYS2

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK        | ADDRESS | LENGTH | FILE | DATE     | PROCSSR |
|--------------|---------|--------|------|----------|---------|
| /IADDRESS/ S | 1061    | 25     |      |          |         |
| /CONTROL/ S  | 1106    | 22     |      |          |         |
| (ANALYS2)    | 36430   | 2      |      |          |         |
| ANALYS2      | 36432   | 102    | LGO  | 80/04/17 | FTN     |

----- SEGMENT - A

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK        | ADDRESS | LENGTH | FILE | DATE     | PROCSSR |
|--------------|---------|--------|------|----------|---------|
| /IADDRESS/ S | 1061    | 25     |      |          |         |
| /CONTROL/ S  | 1106    | 22     |      |          |         |
| (A)          | 36536   | 1      |      |          |         |
| A            | 36537   | 142    | LGO  | 80/04/17 | FTN     |

----- SEGMENT - B

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK        | ADDRESS | LENGTH | FILE | DATE     | PROCSSR |
|--------------|---------|--------|------|----------|---------|
| /IADDRESS/ S | 1061    | 25     |      |          |         |
| /CONTROL/ S  | 1106    | 22     |      |          |         |
| (B)          | 36534   | 1      |      |          |         |
| B            | 36535   | 127    | LGO  | 80/04/17 | FTN     |

----- SEGMENT - C

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK        | ADDRESS | LENGTH | FILE | DATE     | PROCSSR |
|--------------|---------|--------|------|----------|---------|
| /IADDRESS/ S | 1061    | 25     |      |          |         |
| /CONTROL/ S  | 1106    | 22     |      |          |         |
| (C)          | 36534   | 2      |      |          |         |
| C            | 36536   | 62     | LGO  | 80/04/17 | FTN     |

----- SEGMENT - D

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK | ADDRESS | LENGTH | FILE | DATE     | PROCSSR |
|-------|---------|--------|------|----------|---------|
| (D)   | 36701   | 0      |      |          |         |
| D     | 36701   | 206    | LGO  | 80/04/17 | FTN     |

----- SEGMENT - F

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK | ADDRESS | LENGTH | FILE | DATE | PROCSSR |
|-------|---------|--------|------|------|---------|
|-------|---------|--------|------|------|---------|

|     |       |    |     |          |     |
|-----|-------|----|-----|----------|-----|
| (F) | 36664 | 0  |     |          |     |
| F   | 36664 | 65 | LGO | 80/04/17 | FTN |

----- SEGMENT - G

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK | ADDRESS | LENGTH | FILE | DATE     | PROCSSR |
|-------|---------|--------|------|----------|---------|
| (G)   | 36620   | 0      |      |          |         |
| G     | 36620   | 72     | LGO  | 80/04/17 | FTN     |

----- SEGMENT - H

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK | ADDRESS | LENGTH | FILE | DATE     | PROCSSR |
|-------|---------|--------|------|----------|---------|
| (H)   | 36620   | 0      |      |          |         |
| H     | 36620   | 46     | LGO  | 80/04/17 | FTN     |

----- SEGMENT - PLOT

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK        | ADDRESS | LENGTH | FILE | DATE     | PROCSSR |
|--------------|---------|--------|------|----------|---------|
| /IADDRESS/ S | 1061    | 25     |      |          |         |
| /CONTROL/ S  | 1106    | 22     |      |          |         |
| (PLOT)       | 35713   | 2      |      |          |         |
| PLOT         | 35715   | 63     | LGO  | 80/04/17 | FTN     |

----- SEGMENT - PLOTTER

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK        | ADDRESS | LENGTH | FILE | DATE | PROCSSR |
|--------------|---------|--------|------|------|---------|
| /IADDRESS/ S | 1061    | 25     |      |      |         |
| /CONTROL/ S  | 1106    | 22     |      |      |         |
| /FCL.C./ S   | 1130    | 26     |      |      |         |
| /IO.BUF./ S  | 1316    | 227    |      |      |         |

| BLOCK     | ADDRESS | LENGTH | FILE       | DATE     | PROCSSR |
|-----------|---------|--------|------------|----------|---------|
| (PLOTTER) | 36000   | 14     |            |          |         |
| PLOTTER   | 36014   | 373    | LGO        | 80/04/17 | FTN     |
| INPB=     | 36407   | 362    | SL-FORTRAN | 79/12/07 | COMPASS |
| CTL\$SKP  | 36771   | 57     | SL-SYSLIB  | 80/01/09 | COMPASS |

----- SEGMENT - PSEUDO

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK       | ADDRESS | LENGTH | FILE       | DATE     | PROCSSR |
|-------------|---------|--------|------------|----------|---------|
| /FCL.C./ S  | 1130    | 26     |            |          |         |
| (PSEUDO)    | 37050   | 0      |            |          |         |
| /SETCHAR/ G | 37050   | 56     |            |          |         |
| /LANGLEY/ G | 37126   | 25     |            |          |         |
| PSEUDO      | 37153   | 2334   | UL-LRCGOSF | 79/12/10 | COMPASS |
| NUMARG      | 41507   | 114    | UL-LRCGOSF | 79/12/10 | COMPASS |
| WHERE       | 41623   | 31     | UL-LRCGOSF | 79/12/10 | FTN     |
| SYSTEM      | 41654   | 26     | SL-FORTRAN | 79/12/07 | COMPASS |
| ALOG.       | 41702   | 63     | SL-FORTRAN | 79/12/07 | COMPASS |
| EXP.MSG     | 41765   | 16     | SL-FORTRAN | 79/12/07 | COMPASS |
| SINCOS.     | 42003   | 60     | SL-FORTRAN | 79/12/07 | COMPASS |
| SYS=1ST     | 42063   | 63     | SL-FORTRAN | 79/12/07 | COMPASS |
| SYS=AID     | 42146   | 7      | SL-FORTRAN | 79/12/07 | COMPASS |
| XTOI.       | 42155   | 63     | SL-FORTRAN | 79/12/07 | COMPASS |
| CPU.CIO     | 42240   | 13     | SL-SYSLIB  | 79/06/05 | COMPASS |
| CPU.WTO     | 42253   | 17     | SL-SYSLIB  | 79/06/05 | COMPASS |



----- SEGMENT - BSCALE

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK    | ADDRESS | LENGTH | FILE       | DATE     | PROCSSR |
|----------|---------|--------|------------|----------|---------|
| (BSCALE) | 42272   | 0      |            |          |         |
| BSCALE   | 42272   | 506    | UL-LRCGOSF | 79/12/10 | FTN     |
| LEGVAR   | 43000   | 4      | SL-FORTRAN | 79/12/07 | COMPASS |
| EXP.     | 43004   | 72     | SL-FORTRAN | 79/12/07 | COMPASS |
| XTOY.    | 43076   | 31     | SL-FORTRAN | 79/12/07 | COMPASS |

----- SEGMENT - NOTATE

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK       | ADDRESS | LENGTH | FILE       | DATE     | PROCSSR |
|-------------|---------|--------|------------|----------|---------|
| /SETCHAR/ S | 37050   | 56     |            |          |         |
| (NOTATE)    | 42272   | 0      |            |          |         |
| /APOS/      | 42272   | 17     |            |          |         |
| /NOTESSET/  | 42311   | 416    |            |          |         |
| NOTATE      | 42727   | 234    | UL-LRCGOSF | 79/12/10 | FTN     |
| CHARACT     | 43163   | 567    | UL-LRCGOSF | 79/12/10 | FTN     |
| CHNGSET     | 43752   | 63     | UL-LRCGOSF | 79/12/10 | FTN     |
| DECOD1      | 44035   | 7      | UL-LRCGOSF | 79/12/10 | COMPASS |
| NEWSYMR     | 44044   | 172    | UL-LRCGOSF | 79/12/10 | FTN     |

----- SEGMENT - LINPLT

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK    | ADDRESS | LENGTH | FILE       | DATE     | PROCSSR |
|----------|---------|--------|------------|----------|---------|
| (LINPLT) | 42272   | 0      |            |          |         |
| LINPLT   | 42272   | 504    | UL-LRCGOSF | 79/12/10 | FTN     |

|        |       |     |            |          |         |
|--------|-------|-----|------------|----------|---------|
| CIRCLE | 42776 | 165 | UL-LRCGOSF | 79/12/10 | FTN     |
| CNTRLN | 43163 | 134 | UL-LRCGOSF | 79/12/10 | FTN     |
| GRID   | 43317 | 134 | UL-LRCGOSF | 79/12/10 | FTN     |
| PNTPLT | 43453 | 524 | UL-LRCGOSF | 79/12/10 | FTN     |
| SQRT.  | 44177 | 32  | SL-FORTRAN | 79/12/07 | COMPASS |

----- SEGMENT - AXES

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK       | ADDRESS | LENGTH | FILE       | DATE     | PROCSSR |
|-------------|---------|--------|------------|----------|---------|
| /FCL.C./ S  | 1130    | 26     |            |          |         |
| /Q8.IO./ S  | 1156    | 100    |            |          |         |
| /SETCHAR/ S | 37050   | 56     |            |          |         |
| (AXES)      | 44236   | 0      |            |          |         |
| /APOS/      | 44236   | 17     |            |          |         |
| AXES        | 44255   | 1524   | UL-LRCGOSF | 79/12/10 | FTN     |
| NUMBER      | 46001   | 167    | UL-LRCGOSF | 79/12/10 | FTN     |
| ENCODE=     | 46170   | 104    | SL-FORTRAN | 79/12/07 | COMPASS |

----- SEGMENT - DATAOUT

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK        | ADDRESS | LENGTH | FILE | DATE | PROCSSR |
|--------------|---------|--------|------|------|---------|
| /IADDRESS/ S | 1061    | 25     |      |      |         |

| BLOCK       | ADDRESS | LENGTH | FILE | DATE     | PROCSSR |
|-------------|---------|--------|------|----------|---------|
| /CONTROL/ S | 1106    | 22     |      |          |         |
| (DATAOUT)   | 35713   | 1      |      |          |         |
| DATAOUT     | 35714   | 71     | LGO  | 80/04/17 | FTN     |

----- SEGMENT - XOUTPUT

PROGRAM AND BLOCK ASSIGNMENTS.

| BLOCK        | ADDRESS | LENGTH | FILE | DATE     | PROCSSR |
|--------------|---------|--------|------|----------|---------|
| /IADDRESS/ S | 1061    | 25     |      |          |         |
| /CONTROL/ S  | 1106    | 22     |      |          |         |
| (XOUTPUT)    | 36005   | 0      |      |          |         |
| XOUTPUT      | 36005   | 271    | LGO  | 80/04/17 | FTN     |

4.708 CP SECONDS

53700B CM STORAGE USED

## APPENDIX D

ADDRESS FILE FOR INTERACTIVE RUN

|          |               |       |
|----------|---------------|-------|
| IADBLNK= | 46274B,       |       |
| IADIN=   | 35767B,IPIN=  | 167B, |
| IADAN1 = | 36701B,IPAN1= | 206B, |
| IADAN2=  | 36664B,IPAN2= | 65B,  |
| IADPLT = | 46170B,IPPLT= | 104B, |
| IADOUT = | 36005B,IPOUT= | 271B, |

## APPENDIX E

EDITING COMMAND FILE TO GENERATE ADDRESSES FROM LOAD MAP

```

BRIEF
F,,R
IB2
*START
*END
^READ TAPE5
^N
IF$0$
W 80
Z;L$ CM BLANK $;YQM
#####&IADBLNK=#&&&&&&&&&          B,#####
COPY ADRESS
Z;L$ (XINPUT)$;N;YQM
#####&IADIN=&&          , IPIN= #,#####
C$, $B, $
COPY ADRESS
TOP
Z;L$ (D) $;N;YQM
#####&IADAN1=&          , IPAN1=#,#####
C$, $B, $
COPY ADRESS
Z;L$ (F) $;N;YQM
#####&IADAN2=&          , IPAN2=#,#####
C$, $B, $
COPY ADRESS
Z;L$ ENCODE= $;YQM
#####&IADPLT=&          , IPPLT=#,#####
C$, $B, $
COPY ADRESS
Z;L$      XOUTPUT$;YQM
#####&IADOUT=&          , IPOUT=#,#####
C$, $B, $
COPY ADRESS
ELSE
ENDIF
END

```

## APPENDIX F

OUTPUT FILE FOR INTERACTIVE RUN



051063

|        |        |
|--------|--------|
| 035757 | 000167 |
| 036671 | 000206 |
| 036654 | 000065 |
| 051044 | 000017 |
| 035775 | 000271 |

FIELD LENGTH FOR BRANCH -- DATAIN -- IS 037050

FIELD LENGTH FOR BRANCH -- ANALYS1 -- IS 040361

FIELD LENGTH FOR BRANCH -- ANALYS2 -- IS 040223

FIELD LENGTH FOR BRANCH -- PLOT -- IS 046460

FIELD LENGTH FOR BRANCH -- DATAOUT -- IS 037253

INITIAL AA(10) ARRAY

|          |           |          |           |           |
|----------|-----------|----------|-----------|-----------|
| 2.000000 | -7.000000 | 30.00000 | 4.000000  | -10.00000 |
| 2.000000 | 7.000000  | 8.000000 | -8.000000 | -1.000000 |

INITIAL BB(10,15) ARRAY

|           |           |           |           |           |
|-----------|-----------|-----------|-----------|-----------|
| 1.000000  | 2.000000  | 3.000000  | 4.000000  | 5.000000  |
| 6.000000  | 7.000000  | 8.000000  | 9.000000  | 10.00000  |
| 11.00000  | 12.00000  | 13.00000  | 14.00000  | 15.00000  |
| 1.500000  | 2.500000  | 3.500000  | 4.500000  | 5.500000  |
| 6.500000  | 7.500000  | 8.500000  | 9.500000  | 10.50000  |
| 11.50000  | 12.50000  | 13.50000  | 14.50000  | 15.50000  |
| -1.000000 | -2.000000 | -3.000000 | -4.000000 | -5.000000 |
| -6.000000 | -7.000000 | -8.000000 | -9.000000 | -10.00000 |
| -11.00000 | -12.00000 | -13.00000 | -14.00000 | -15.00000 |
| 0.        | 1.000000  | 2.000000  | 3.000000  | 4.000000  |
| 5.000000  | 6.000000  | 7.000000  | 8.000000  | 9.000000  |
| 10.00000  | 11.00000  | 12.00000  | 13.00000  | 14.00000  |
| .5000000  | 1.500000  | 2.500000  | 3.500000  | 4.500000  |
| 5.500000  | 6.500000  | 7.500000  | 8.500000  | 9.500000  |
| 10.50000  | 11.50000  | 12.50000  | 13.50000  | 14.50000  |
| 0.        | -1.000000 | -2.000000 | -3.000000 | -4.000000 |
| -5.000000 | -6.000000 | -7.000000 | -8.000000 | -9.000000 |
| -10.00000 | -11.00000 | -12.00000 | -13.00000 | -14.00000 |
| 0.        | 0.        | 1.000000  | 2.000000  | 3.000000  |
| 4.000000  | 5.000000  | 6.000000  | 7.000000  | 8.000000  |
| 9.000000  | 10.00000  | 11.00000  | 12.00000  | 13.00000  |
| 0.        | 0.        | .5000000  | 1.500000  | 2.500000  |
| 3.500000  | 4.500000  | 5.500000  | 6.500000  | 7.500000  |
| 8.500000  | 9.500000  | 10.50000  | 11.50000  | 12.50000  |
| 13.50000  | 0.        | 0.        | -1.000000 | -2.000000 |

|            |           |            |            |            |
|------------|-----------|------------|------------|------------|
| -3.000000  | -4.000000 | -5.000000  | -6.000000  | -7.000000  |
| -8.000000  | -9.000000 | -10.000000 | -11.000000 | -12.000000 |
| -13.000000 | 1.000000  | 0.         | 1.000000   | 0.         |
| 1.000000   | 0.        | 1.000000   | 0.         | 1.000000   |
| 0.         | 1.000000  | 0.         | 1.000000   | 0.         |

INITIAL CC(15,15) ARRAY

|            |           |            |           |            |
|------------|-----------|------------|-----------|------------|
| 1.000000   | 1.000000  | -2.000000  | 3.000000  | -4.000000  |
| 5.000000   | -6.000000 | 7.000000   | -8.000000 | 9.000000   |
| -10.000000 | 11.000000 | -12.000000 | 13.000000 | -14.000000 |
| 15.000000  | 2.000000  | 4.000000   | 3.000000  | 6.000000   |
| 5.000000   | 8.000000  | 7.000000   | 10.000000 | 9.000000   |
| 12.000000  | 11.000000 | 14.000000  | 13.000000 | 16.000000  |
| 15.000000  | 0.        | 1.000000   | 0.        | 2.000000   |
| 0.         | 3.000000  | 0.         | 4.000000  | 0.         |
| 5.000000   | 0.        | 6.000000   | 0.        | 7.000000   |
| 0.         | 0.        | 0.         | 1.000000  | 0.         |
| 0.         | 2.000000  | 0.         | 0.        | 3.000000   |
| 0.         | 0.        | 4.000000   | 0.        | 0.         |
| 5.000000   | 1.000000  | 0.         | 0.        | -1.000000  |
| 0.         | 0.        | 2.000000   | 0.        | 0.         |
| -2.000000  | 0.        | 0.         | 3.000000  | 0.         |
| 0.         | -9.000000 | 0.         | -8.000000 | 0.         |
| -7.000000  | 0.        | -6.000000  | 0.        | -5.000000  |
| 0.         | -4.000000 | 0.         | -3.000000 | 0.         |
| -2.000000  | 1.000000  | 2.000000   | 1.000000  | 2.000000   |
| 1.000000   | 2.000000  | 1.000000   | 2.000000  | 1.000000   |
| 2.000000   | 1.000000  | 2.000000   | 1.000000  | 2.000000   |
| 1.000000   | -1.000000 | 2.000000   | -1.000000 | 2.000000   |
| -1.000000  | 2.000000  | -1.000000  | 2.000000  | 2.000000   |
| -1.000000  | 2.000000  | -1.000000  | 2.000000  | -1.000000  |
| 2.000000   | 0.        | 0.         | 0.        | 1.000000   |
| 0.         | 0.        | 0.         | -1.000000 | 0.         |
| 0.         | 0.        | 1.000000   | 0.        | 0.         |
| 0.         | 0.        | 0.         | 0.        | 0.         |
| 0.         | 0.        | 0.         | 0.        | 0.         |
| 0.         | 0.        | 0.         | 0.        | 0.         |
| 0.         | 0.        | 0.         | 0.        | 0.         |
| 1.000000   | 0.        | 0.         | 0.        | 0.         |
| 0.         | 0.        | 0.         | 0.        | 0.         |
| 0.         | 0.        | 0.         | 0.        | 1.000000   |
| 0.         | 0.        | 0.         | 0.        | 0.         |
| 0.         | 0.        | 0.         | 0.        | 0.         |
| 0.         | 0.        | 0.         | 0.        | 0.         |
| 0.         | 0.        | 0.         | 1.000000  | 0.         |
| 0.         | 0.        | 0.         | 0.        | 0.         |
| 0.         | 0.        | 0.         | 0.        | 0.         |
| 0.         | 0.        | 0.         | 0.        | 0.         |
| 0.         | 0.        | 1.000000   | 0.        | 0.         |
| 0.         | 0.        | 0.         | 0.        | 0.         |
| 0.         | 0.        | 0.         | 0.        | 0.         |
| 0.         | 0.        | 0.         | 0.        | 0.         |
| 0.         | 1.000000  | 0.         | 0.        | 0.         |
| 0.         | 0.        | 0.         | 0.        | 0.         |
| 0.         | 0.        | 0.         | 0.        | 0.         |
| 1.000000   | 0.        | 0.         | 0.        | 0.         |

| TIME  | DET        | F(T)      |
|-------|------------|-----------|
| 0.    | -.2100E-13 | .8637E+14 |
| .2000 | .2700E-10  | .2704E+17 |
| .4000 | -.4958E-09 | .2800E+15 |
| .6000 | -.4696E-07 | .8174E+16 |
| .8000 | -.1171E-06 | .4611E+17 |
| 1.000 | .7658E-07  | .9089E+17 |
| 1.200 | -.6696E-10 | .4738E+12 |
| 1.400 | -.4338E-06 | .1323E+18 |
| 1.600 | .1966E-06  | .7200E+14 |
| 1.800 | .6426E-06  | .8781E+17 |
| 2.000 | .1055E-06  | .3308E+17 |
| 2.200 | .5434E-07  | .1338E+14 |
| 2.400 | -.7532E-04 | .2013E+18 |
| 2.600 | .6969E-05  | .2403E+18 |
| 2.800 | .1902E-05  | .6555E+17 |

## **APPENDIX G**

### **OUTPUT FILE FOR BATCH RUN**

FIELD LENGTH FOR BRANCH -- DATAIN -- IS 052166

FIELD LENGTH FOR BRANCH -- ANALYS1 -- IS 053107

FIELD LENGTH FOR BRANCH -- ANALYS2 -- IS 052546

FIELD LENGTH FOR BRANCH -- PLOT -- IS 052251

FIELD LENGTH FOR BRANCH -- DATAOUT -- IS 052251

INITIAL AA(10) ARRAY

|          |           |          |           |           |
|----------|-----------|----------|-----------|-----------|
| 2.000000 | -7.000000 | 30.00000 | 4.000000  | -10.00000 |
| 2.000000 | 7.000000  | 8.000000 | -8.000000 | -1.000000 |

INITIAL BB(10,15) ARRAY

|           |           |           |           |           |
|-----------|-----------|-----------|-----------|-----------|
| 1.000000  | 2.000000  | 3.000000  | 4.000000  | 5.000000  |
| 6.000000  | 7.000000  | 8.000000  | 9.000000  | 10.00000  |
| 11.00000  | 12.00000  | 13.00000  | 14.00000  | 15.00000  |
| 1.500000  | 2.500000  | 3.500000  | 4.500000  | 5.500000  |
| 6.500000  | 7.500000  | 8.500000  | 9.500000  | 10.50000  |
| 11.50000  | 12.50000  | 13.50000  | 14.50000  | 15.50000  |
| -1.000000 | -2.000000 | -3.000000 | -4.000000 | -5.000000 |
| -6.000000 | -7.000000 | -8.000000 | -9.000000 | -10.00000 |
| -11.00000 | -12.00000 | -13.00000 | -14.00000 | -15.00000 |
| 0.        | 1.000000  | 2.000000  | 3.000000  | 4.000000  |
| 5.000000  | 6.000000  | 7.000000  | 8.000000  | 9.000000  |
| 10.00000  | 11.00000  | 12.00000  | 13.00000  | 14.00000  |
| .5000000  | 1.500000  | 2.500000  | 3.500000  | 4.500000  |
| 5.500000  | 6.500000  | 7.500000  | 8.500000  | 9.500000  |
| 10.50000  | 11.50000  | 12.50000  | 13.50000  | 14.50000  |
| 0.        | -1.000000 | -2.000000 | -3.000000 | -4.000000 |
| -5.000000 | -6.000000 | -7.000000 | -8.000000 | -9.000000 |
| -10.00000 | -11.00000 | -12.00000 | -13.00000 | -14.00000 |
| 0.        | 0.        | 1.000000  | 2.000000  | 3.000000  |
| 4.000000  | 5.000000  | 6.000000  | 7.000000  | 8.000000  |
| 9.000000  | 10.00000  | 11.00000  | 12.00000  | 13.00000  |
| 0.        | 0.        | .5000000  | 1.500000  | 2.500000  |
| 3.500000  | 4.500000  | 5.500000  | 6.500000  | 7.500000  |
| 8.500000  | 9.500000  | 10.50000  | 11.50000  | 12.50000  |
| 13.50000  | 0.        | 0.        | -1.000000 | -2.000000 |
| -3.000000 | -4.000000 | -5.000000 | -6.000000 | -7.000000 |
| -8.000000 | -9.000000 | -10.00000 | -11.00000 | -12.00000 |
| -13.00000 | 1.000000  | 0.        | 1.000000  | 0.        |
| 1.000000  | 0.        | 1.000000  | 0.        | 1.000000  |
| 0.        | 1.000000  | 0.        | 1.000000  | 0.        |

INITIAL CC(15,15) ARRAY

|           |           |           |           |           |
|-----------|-----------|-----------|-----------|-----------|
| 1.000000  | 1.000000  | -2.000000 | 3.000000  | -4.000000 |
| 5.000000  | -6.000000 | 7.000000  | -8.000000 | 9.000000  |
| -10.00000 | 11.00000  | -12.00000 | 13.00000  | -14.00000 |
| 15.00000  | 2.000000  | 4.000000  | 3.000000  | 6.000000  |
| 5.000000  | 8.000000  | 7.000000  | 10.00000  | 9.000000  |
| 12.00000  | 11.00000  | 14.00000  | 13.00000  | 16.00000  |
| 15.00000  | 0.        | 1.000000  | 0.        | 2.000000  |
| 0.        | 3.000000  | 0.        | 4.000000  | 0.        |
| 5.000000  | 0.        | 6.000000  | 0.        | 7.000000  |
| 0.        | 0.        | 0.        | 1.000000  | 0.        |
| 0.        | 2.000000  | 0.        | 0.        | 3.000000  |
| 0.        | 0.        | 4.000000  | 0.        | 0.        |
| 5.000000  | 1.000000  | 0.        | 0.        | -1.000000 |
| 0.        | 0.        | 2.000000  | 0.        | 0.        |
| -2.000000 | 0.        | 0.        | 3.000000  | 0.        |
| 0.        | -9.000000 | 0.        | -8.000000 | 0.        |
| -7.000000 | 0.        | -6.000000 | 0.        | -5.000000 |
| 0.        | -4.000000 | 0.        | -3.000000 | 0.        |
| -2.000000 | 1.000000  | 2.000000  | 1.000000  | 2.000000  |
| 1.000000  | 2.000000  | 1.000000  | 2.000000  | 1.000000  |
| 2.000000  | 1.000000  | 2.000000  | 1.000000  | 2.000000  |
| 1.000000  | -1.000000 | 2.000000  | -1.000000 | 2.000000  |
| -1.000000 | 2.000000  | -1.000000 | 2.000000  | 2.000000  |
| -1.000000 | 2.000000  | -1.000000 | 2.000000  | -1.000000 |
| 2.000000  | 0.        | 0.        | 0.        | 1.000000  |
| 0.        | 0.        | 0.        | -1.000000 | 0.        |
| 0.        | 0.        | 1.000000  | 0.        | 0.        |
| 0.        | 0.        | 0.        | 0.        | 0.        |
| 0.        | 0.        | 0.        | 0.        | 0.        |
| 0.        | 0.        | 0.        | 0.        | 0.        |
| 0.        | 0.        | 0.        | 0.        | 0.        |
| 1.000000  | 0.        | 0.        | 0.        | 0.        |
| 0.        | 0.        | 0.        | 0.        | 0.        |
| 0.        | 0.        | 0.        | 0.        | 1.000000  |
| 0.        | 0.        | 0.        | 0.        | 0.        |
| 0.        | 0.        | 0.        | 0.        | 0.        |
| 0.        | 0.        | 0.        | 1.000000  | 0.        |
| 0.        | 0.        | 0.        | 0.        | 0.        |
| 0.        | 0.        | 0.        | 0.        | 0.        |
| 0.        | 0.        | 0.        | 0.        | 0.        |
| 0.        | 0.        | 0.        | 0.        | 0.        |
| 0.        | 0.        | 0.        | 0.        | 0.        |
| 0.        | 1.000000  | 0.        | 0.        | 0.        |
| 0.        | 0.        | 0.        | 0.        | 0.        |
| 0.        | 0.        | 0.        | 0.        | 0.        |
| 0.        | 0.        | 0.        | 0.        | 0.        |
| 1.000000  | 0.        | 0.        | 0.        | 0.        |

TIME

DET

F(T)

|       |            |           |
|-------|------------|-----------|
| 0.    | -.2100E-13 | .8637E+14 |
| .2000 | .2700E-10  | .2704E+17 |
| .4000 | -.4958E-09 | .2800E+15 |
| .6000 | -.4696E-07 | .8174E+16 |

|       |            |           |
|-------|------------|-----------|
| .8000 | -.1171E-06 | .4611E+17 |
| 1.000 | .7658E-07  | .9089E+17 |
| 1.200 | -.6696E-10 | .4738E+12 |
| 1.400 | -.4338E-06 | .1323E+18 |
| 1.600 | .1966E-06  | .7200E+14 |
| 1.800 | .6426E-06  | .8781E+17 |
| 2.000 | .1055E-06  | .3308E+17 |
| 2.200 | .5434E-07  | .1338E+14 |
| 2.400 | -.7532E-04 | .2013E+18 |
| 2.600 | .6969E-05  | .2403E+18 |
| 2.800 | .1902E-05  | .6555E+17 |

## APPENDIX H

CONTROL CARD FILE TO EXECUTE PROGRAM FROM RELOCATABLE BINARY



```

.PROC, PROSAM, SAMMAP.
REWIND, SAMMAP, SAMABS, LOADIR.
GET, LGO=SAMBIN.
GET, EDILOAD, BATLOAD.
GET, TAPE5=SAMPDAT.
ATTACH, FTMMLIB, LRCGOSF/UN=LIBRARY.
XEDIT, BATLOAD, L=NOPRINT./CALL EDILOAD/
SEGLOAD, B=SAMABS, I=LOADIR, LO=DT.
LDSET, LIB=FTMMLIB/LRCGOSF, MAP=/SAMMAP, PRESET=ZERO.
LOAD, LGO.
NOGO.
XEDIT, SAMMAP, L=NOPRINT, W=132./CALL EDIMAP/
SAMABS.
REVERT.*** END LOAD SEQUENCE FOR SAMABS ***
EXIT.
REVERT, ABORT.*** ERROR IN LOAD SEQUENCE ***

```

Note: Binary file SAMBIN was generated using the "static" option on the FTM control card in order to allow for the dynamic data storage allocation:

```
FTN, I=SAMPLE, B=SAMBIN, STATIC.
```

The "static" option is unnecessary if employing only variable loading techniques, without dynamic storage allocation.

## REFERENCES

1. Tiffany, Sherwood H.; Newsom, Jerry R.: Some Programming Techniques for Increasing Program Versatility and Efficiency on CDC Equipment. NASA CR-3033, August, 1978.
2. Control Data Corporation: Cyber Loader Version 1 Reference Manual. Publication No. 60429800, 1979.
3. Langley Research Center: Graphic Output System Users Guide. NASA Langley Research Center Computer Programming Manual, Vol. IV, 1979.

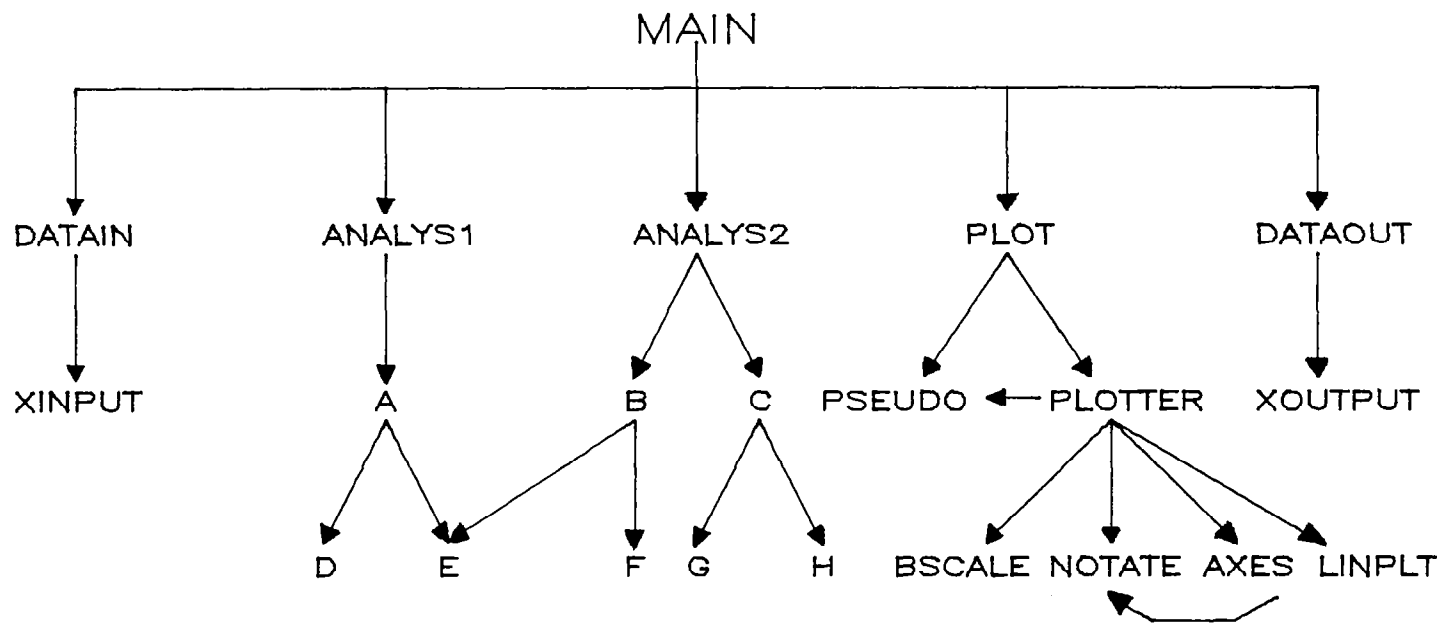


Figure 1.- Calling sequence diagram.

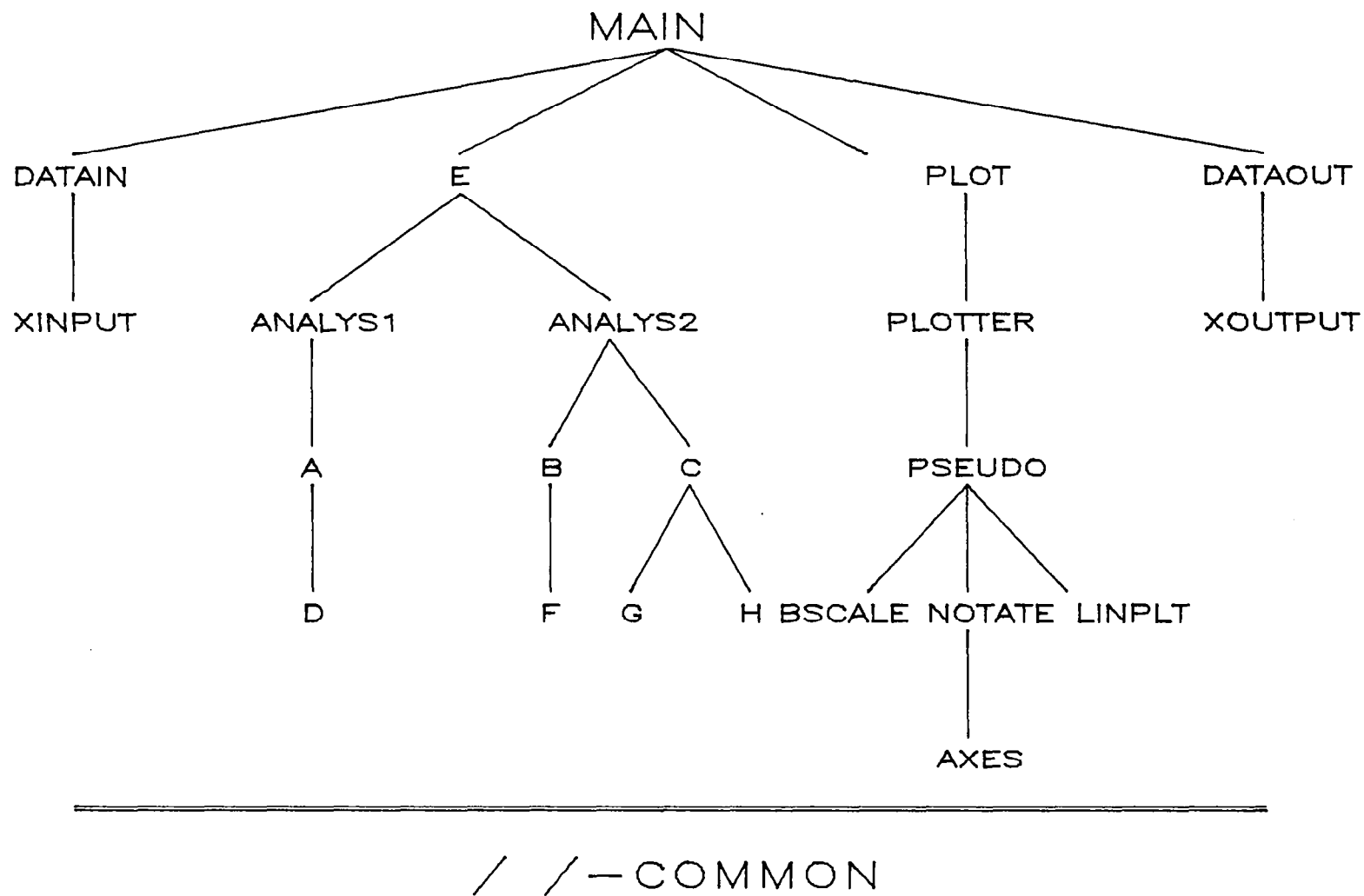


Figure 2.- Branching diagram for interactive run.

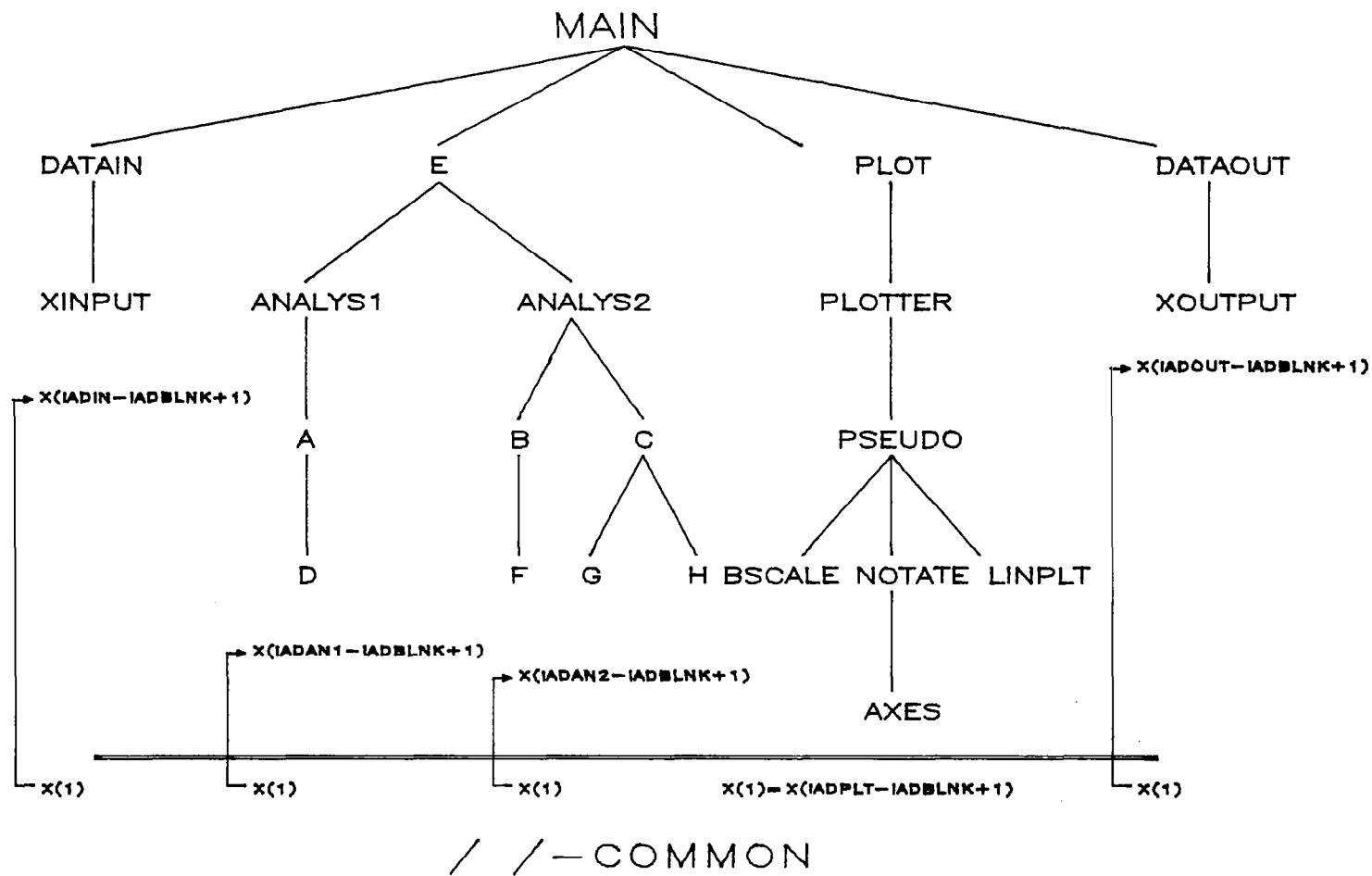
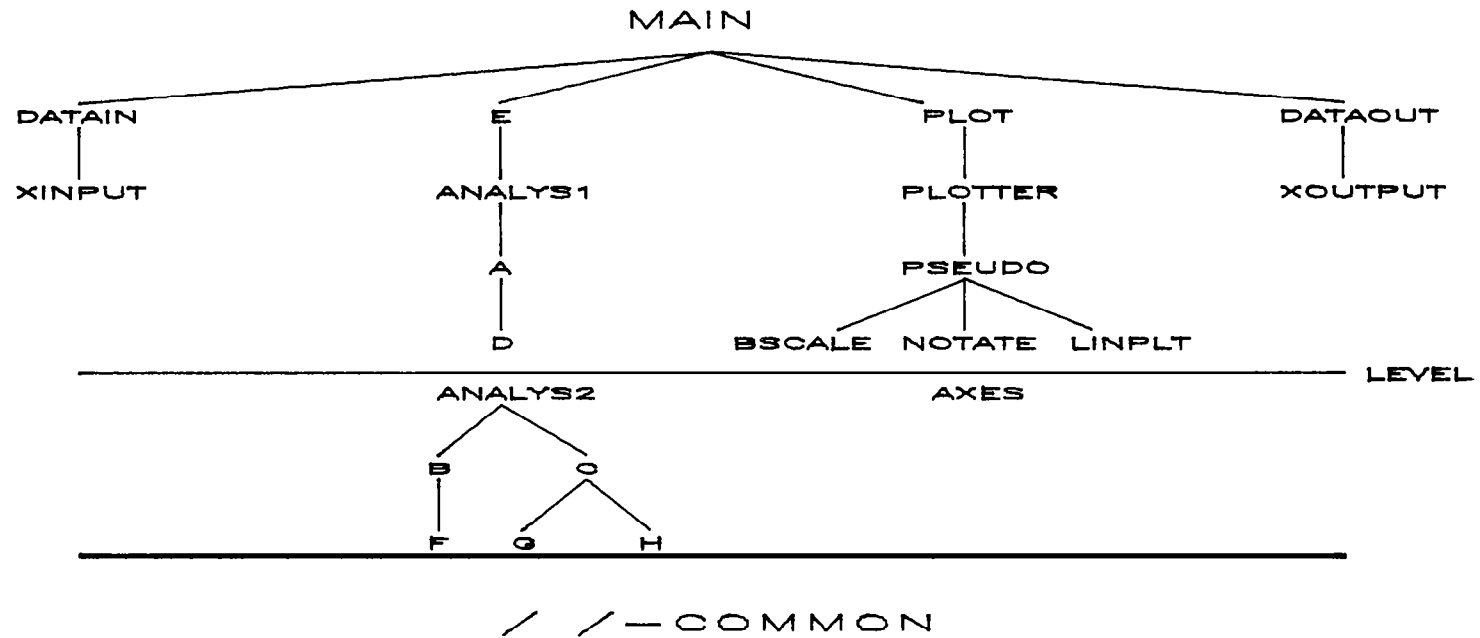
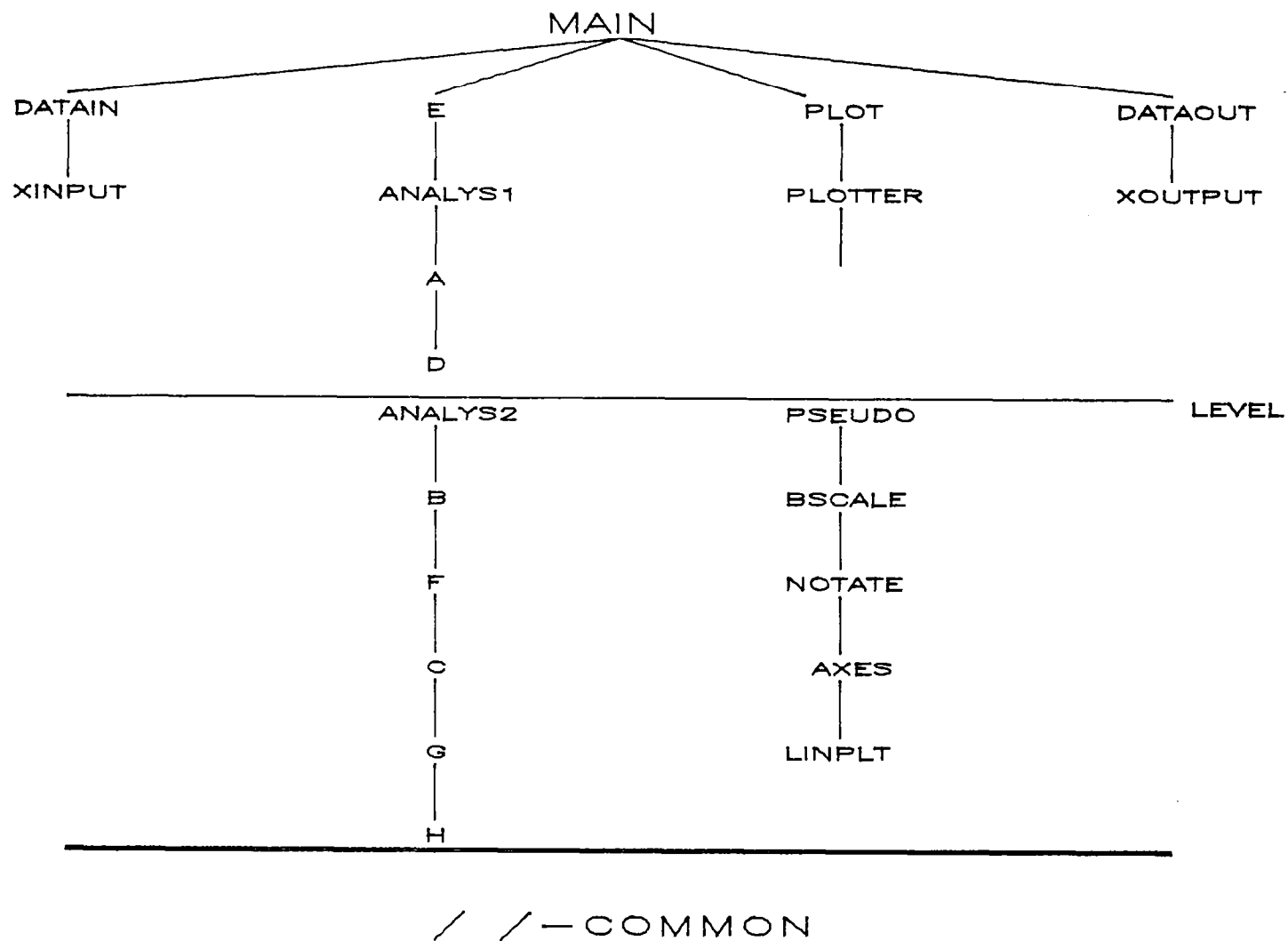


Figure 3.- Beginning data storage addresses for interactive run.



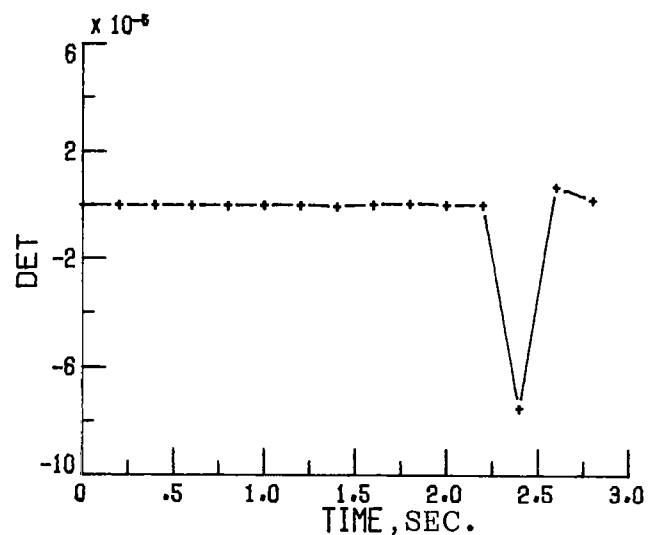
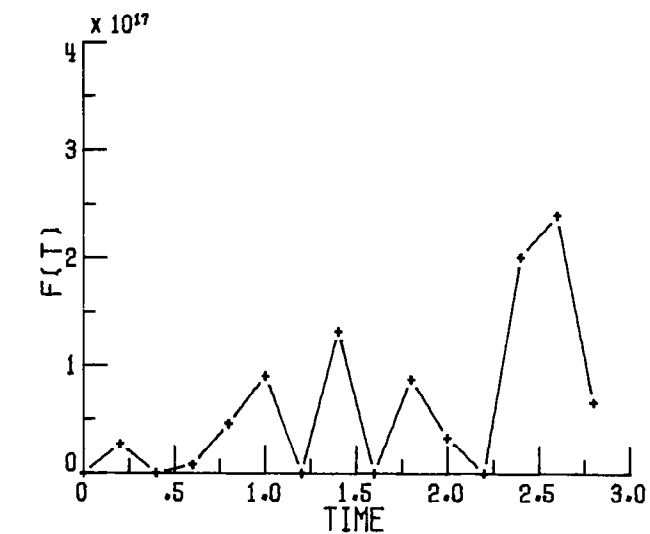
(a) Branch diagram 1

Figure 4.- Possible branching diagrams for batch run.



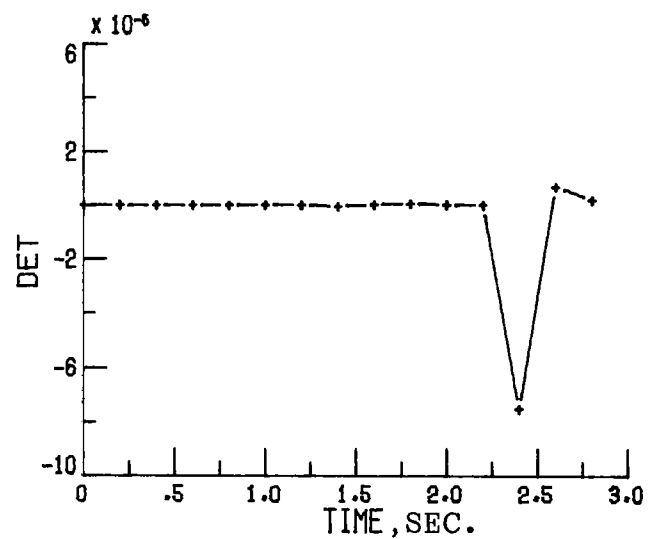
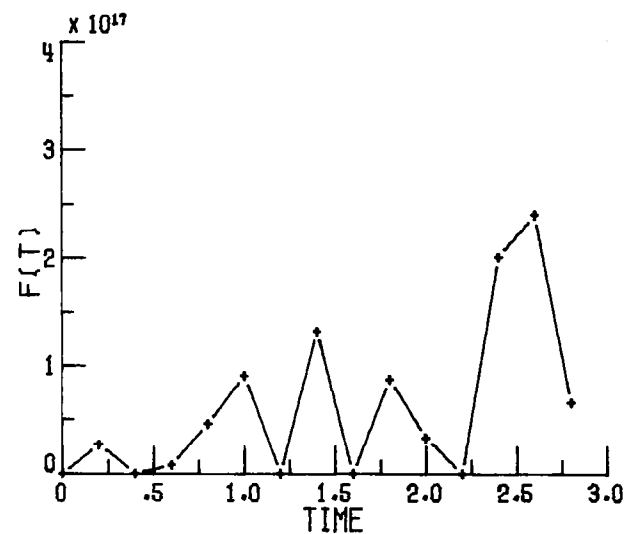
(b) Branch diagram 2

Figure 4.- Concluded.



PLOTS VS. TIME FOR SAMPLE PROGRAM - INTERACTIV

(a) Interactive run



PLOTS VS. TIME FOR SAMPLE PROGRAM - BATCH

(b) Batch run

Figure 5.- Sample plots from interactive and batch runs.



|  |  |                             |   |  |  |
|--|--|-----------------------------|---|--|--|
| 1. Report No.<br>NASA CR-3315  |  | 2. Government Accession No. |   | 3. Recipient's Catalog No.                                 |  |
| 4. Title and Subtitle<br>SEGMENTATION, DYNAMIC STORAGE, AND VARIABLE LOADING<br>ON CDC EQUIPMENT   |  |                             |   | 5. Report Date<br>September 1980                           |  |
|  |  |                             |   | 6. Performing Organization Code                            |  |
| 7. Author(s)<br>Sherwood H. Tiffany  |  |                             |   | 8. Performing Organization Report No.                      |  |
| 9. Performing Organization Name and Address<br>Kentron International, Inc.<br>Hampton Technical Center<br>3221 North Armistead Avenue<br>Hampton, VA 23666   |  |                             |   | 10. Work Unit No.  |  |
|  |  |                             |   | 11. Contract or Grant No.<br>NAS1-16000                    |  |
| 12. Sponsoring Agency Name and Address<br>National Aeronautics and Space Administration<br>Washington, DC 20546  |  |                             |   | 13. Type of Report and Period Covered<br>Contractor Report |  |
|  |  |                             |   | 14. Sponsoring Agency Code                                 |  |
| 15. Supplementary Notes<br>Technical Monitor: William M. Adams, Jr.<br>Topical Report  |  |                             |   |  |  |
| 16. Abstract<br><br>Techniques for varying the segmented load structure of a program and for varying the dynamic storage allocation, depending upon whether a batch type or interactive type run is desired, are explained and demonstrated. All changes are based on a single data input to the program. The techniques involve code within the program to suppress scratch pad input/output (IØ) for a "batch" run or translate the in-core data storage area from blank common to the end-of-code+1 address of a particular segment for an "interactive" run; automatic editing of the segload directives prior to loading, based upon data input to the program, to vary the structure of the load for "interactive" and "batch" runs; and automatic editing of the load map to determine the initial addresses for in-core data storage for an "interactive" run. |  |                             |   |  |  |
| 17. Key Words (Suggested by Author(s))<br>Automatic variable loading<br>Segmentation<br>Dynamic Storage allocation   |  |                             | 18. Distribution Statement<br>Unclassified - Unlimited<br><br>Subject Category 61 |  |  |
| 19. Security Classif. (of this report)<br>Unclassified   | 20. Security Classif. (of this page)<br>Unclassified | 21. No. of Pages<br>70      | 22. Price<br>A04  |  |  |